DRAGON USER

The independent Dragon magazine

May 1988

Contents

Letters 2
Motorbiking ... Booting DragonDOS ... Dragon Professionals ...

Crossword 3
Crossword six with tapes to win

News 4
Quickbeam quits ... new Pulser product ... Orange Software: new house launches ... Comms port ... Cardiff Airport ... Martin Vernon found alive ... new DUDE organiser ...

Pamcodes 6
Pam D'Arcy continues with the sixth part of her introduction to machine code.

Dragonsoft 8, 9, 11, 14
Pyradventure from Dragonfire and Orange, *Crazy Foota 2* and *3* from Computape and Orange, *Space Trek* and *Reversi* from R & AJ Preston, *Printer Control* from MacGowan Software, *Picture Maker* from John Penn Software.

Tandy discs 10
Eric Hall looks closely at the format of Tandy discs and how they can be exploited by Dragon users.

Point your Dragon 12
Nigel Mason describes how to incorporate a mouse-controlled arrow into the Dragon.

Winners and losers 15
Gordon Lee discusses the solutions to the December competition.

Making tracks 16
Philip G. Scott suggests a way to get more storage space out of your discs.

Review: Cumana disc drives 18
Ken Smith describes his Cumana drives and the DOSs which operate with them.

Write: ADVENTURE 19
This month, Pete Gerrard concentrates on getting it all.

Adventure trail 20
Pete Gerrard, Dimli and Strombrigner get involved in the *Cricklewood Incident*.

Competition 22
Gordon Lee gets hung up on number series — will they let him take his *Mensa* test?

Dragon Answers 24
Declaring arrays in Basic ... Serial numbers on the Dragon ... Merging Basic ...

Show Report 24
Grahame Smith of Orange Software does the participant bit and reports from Cardiff Airport.

Editorial

ONCE again — don't forget the Ossett Show: April 30th, just south of Leeds off the M1, more information from John Penn Discount Software on 04203 5970. Easter has made the Show a bit late in booking, so we don't have any more definite information at the Dragon User office in time for this issue.

What we do have is the biggest influx of software reviews for some time, so I have turned over all the spare half pages and cubby holes to *Dragon Soft*, which is why you will keep coming across it. More all-new software next month, I hope.

This month's major piece of news is 'Quickbeam quits — official.' In fact, Computape will be handling Quickbeam's former list, but there will be no new software from QB. Computape are, as we said previously, moving their establishment soon, but if you have any difficulty getting through to the old number — keep trying. They haven't shifted yet, they're just very busy.

Next month's big news is uuummmpphgh. (*You'll just have to wait and see — love, the Editor's computer.*).

Telephone number
(All departments)
437-4343

Editor
HELEN ARMSTRONG

Production Editor
DRAGON EDITORIAL

Administration
CAROL FRITH

Advertisement Manager
DRAGON EDITORIAL

Marketing Manager
HELEN PERRY

Managing Editor
PETER KANE

Publishing Director
JENNY IRELAND

Subscriptions
UK £14 for 12 issues
Overseas (surface) £20 for 12 issues
ISSN 0265-177. Telex: 296275
Dragon User, 12/13 Little Newport Street, London WC2H 7PP
US address: c/o Business Press International, 205 East 42nd St, New York, NY 10017
Published by Sunshine Publications 1988

Typesetting and Production by Artset Limited, London NW1.
Printed by Headley Brothers Ltd. Ashford, Kent
Registered at the Post Office as a newspaper.
Dragon and its logo are trademarks of Eurohard Ltd.

How to submit articles

The quality of the material we can publish in *Dragon User* each month will, to a very great extent depend on the quality of the discoveries that you can make with your Dragon. The Dragon computer was launched on to the market with a powerful version of Basic, but with very poor documentation.

Articles which are submitted to *Dragon User* for publication should not be more than 3000 words long. All submissions should be typed. Please leave wide margins and a double space between each line. Programs should, whenever possible, be computer printed on plain white paper and be accompanied by a tape of the program.

We cannot guarantee to return every submitted article or program, so please keep a copy. If you want to have your program returned you must include a stamped addressed envelope.

Letters

This is your chance to air your views — send your tips, compliments and complaints to Letters Page, *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

FLEXible approach

PLEASE could you tell me where I can obtain the FLEX *Advanced Programmer's Guide*. Also, could anyone tell me whether there is a FLEX users group, similar to the OS-9 Users' Group.

P D Smith
University Hall
Birchwood
Penylan
Cardiff
CF2 5YB

Motorbike repair

WE found that during typing *Motorbiking* by Richard Boryna from the listing in DU November 1987, that some co-ordinates for the LINE commands in lines 390 and 2670 were missing. These should read:

```
390 LINE(167-BM,191)-(167-BM,176),PSET:LINE(167-BM,176)-(167-BM-29,191),PSET:PAINT(165-BM,190)
2670 LINE(180,191)-(180,176),PSET:LINE(180,176)-(209,191),PSET:PAINT(190,190)
```

To make the program suitable for DragonDOS, type the following lines:

```
80 CLS:POKE&HFF48,0
140 CLEAR300,32652
150 FOR I=0 TO 113:READ A$:POKE32653+I,VAL("&H"+A$):NEXT
155 DATA 1A,10,CC,7F,A3,FD,01,0D,1C,EF,39
156 DATA 1A,10,CC,C7,00,FD,01,0D,1C,EF,39
300 EXEC32653
2240 IF I$="Q" THEN EXEC32664:END
```

Lilne 80 clears the disc motor timer before IRQ is disabled. Line 155 contains machine code to redirect the IRQ interrupt to the SOUND routine while line 156 restores the normal DOS IRQ vector when you type Q to quit the game.

Should you wish to try this version using cassette tape, then line 156 should be edited to read:

```
156 DATA 1A,10,CC,9D,3D,FD,01,0D,1C,EF,39
```

PS — Don't forget the 'repair' line 2335 from the December issue — 2335 S=0:P=0:X=6.

Bob and Ian Thomas
5 Evergreen Close
Hempstead
Gillingham
Kent
ME7 3QY

Every month we will be shelling out a game or two, courtesy of our supplies, to the reader/s who send the most interesting or entertaining letters. So send us your hints and your opinions, send us your hi-scores and suggestions. Send us your best Dragon stories. What d'you think we are, mind readers?!

EXTRA PUFF

Let's put our money where our mouths are

THE Dragon market is small and game sales are dwindling. Since the publisher is in business to make a profit, it makes sense to target the largest section of the market. This is necessary to guarantee the recovery of advertising costs, duplication costs for a few thousand cassettes, and the high initialisation and printing cost of the inlay cards. If the game rally sells well then there is maybe a healthy profit, and never forget, this is what keeps them in business.

There is an alternative: if the smaller market is willing to give a larger profit margin per game then this could offset low sales. That smaller market would be expected to be a bit more choosy, because when you pay more you expect the best. Ask yourself: if the Dragon version of say *Manic Miner* is better than the Spectrum version (which it was), am I prepared to pay more for it?

In the US there are two new games which will run happily on the Dragon. These are *Popeye the Sailor Man* and *Marble Maze*, excellent programs of which I have first hand knowledge. You can get them from Tandy in the States, and by the time you have paid for postage, packing and import duty they cost about £30 each. If they were produced here without all the eye-catching cards they could retail at about half that. I suggest you all write to the software houses and tell them you would be prepared to pay ten to fifteen pounds for such a program. Point out that you do not need fancy inlays, a black and white instruction sheet would do. It would then be up to us to put our money where our mouths are. We do need continued software support but we will not get it if people go broke trying to supply us.

Ken Smith, 33 Glack Road, Deal, Kent CT14 9ND

THIS is an alternative and in my view accurate point of view to the oft expressed "if only dealers would sell their wares at rock bottom prices, everyone would buy them and we would all be ok". There is a limit to how far prices can be slashed before the very act of selling them makes a loss.

Obliquely if not directly related to Ken's theme are Dave Hitchman's comments on our news page this month — see page 4.

Boot is patched

Having entered Julian Osbourne's *Auto Boot* program in the October *Dragon User*, I discovered that it would not run correctly with DragonDOS 4.0.

I enclose the folloiwng corrections which allow the program to work with DOS 4.0. The revised program works with both the BOOT command and the startup boot routine available with DOS 4.0.

Insert LDA £$22 after FCC /OS/ line; change JMP $D4A4 to JMP $D4B7 and change CMPA £19 to CMPA £21.

The first part of the assembler source therefore reads:

```
ORG 9728
START FCC /OS/
LDA £$22
LDX £FNAME
STX 166
JMP $D4B7
FNAME FCC 34,/
  MENU.BAS/,34,0
```

When using the loader program the following lines should be used instead of the published ones:

```
60 DATA 4F,53,86,22,8E,26,0C,9F,A6,7E,D4,B7
70 DATA 22,4D,45,4E,55,2E,42,41,53,22,00,4F
80 DATA 0F,EC,86,03,97,ED,8E,26,00,34,10,9F
90 DATA EE,BD,C1,01,35,10,30,89,01,00,0C,ED
100 DATA 96,ED,81,15,25,EB,7E,83,71,39,**
```

In the main body of the text the save routine becomes SAVE "BOOT.BIN",&H2600,&H2639,&H2617 and the EXEC address becomes &H2617.

Note that if the filename to be used is longer than MENU.BAS then the CMPA £21 instruction above and the SAVE command should be increased appropriately.

Richard Christon
10 St. Oswald's Close
Thirsk, N. Yorks. YO7 1JX

True Professionals

ON the subject of the Dragon Professional, readers may be interested to know that at least one prototype was actually built. A report on a bench-test was printed in *Personal Computer World*. I am sorry that I am unable to give an exact reference, as I have lost the magazine concerned, but I think it was early in 1984. As surmised in the March *Dragon*

User, the machine was produced just before Dragon Data Ltd. was taken over by GEC.

The Professional was, if I remember correctly, equivalent to a Dragon 64, with twin disk drives mounted in the same box. The general layout was similar to an Apple 2. The specification was somewhat disappointing for a professional machine — for example, the 32-column screen display was retained.

The reviewer complained that the machine stopped working after an hour, as it overheated. The conclusion was that further technical development was needed, and that, perhaps, Dragon Data had lost too much credibility to be able to market the machine successfully.

It seems amazine that no review appeared in *Dragon User*. Does this tell us anything we don't already know about Dragon Data Ltd.'s business sense?

Frank Waterland
56 Roe Lane
Kingsbury
London NW9 9BD

WITH regard to the 'Professional', there were indeed 'finished' models up and running at the last 6809 Show, just as Dragon Data collapsed. GEC produced a business pack at that show with the Professional labelled D6009. Their literature was stamped 'Preliminary Product Information' and was produced at the same time as the Touchmaster pad labelled D6011, also stamped accordingly.

The Professional was described as offering 'in one compact unit a powerful computing package utilising the 6809 microprocessor'. With GEC Dragon OS-9 software, the Professional had a wide range of problem solving applications for professional and small business users.

Features included an integral Sony 3.5in disc drive of 500Kbytes unformatted, an integral modem with 1200/75 baud Prestel mode and 600/600, baud machine/machine maode, 64K ram, 16K rom Basic interpreter, 8K rom modem controller, five graphics modes: 32 characters x 16 lines screen with nine colours, 64 x 32 with nine colours, 128 x 96 with two sets of four colours, 128 x 192 with two sets of four colours and 256 x 192 with two sets of two colours; 8 octave sound synthesiser with three independent voices; full travel keyboard guaranteed 20 million depressions ((*That reminds me. I have to ring my tax person — Ed.*)) and internal switched mode power supply. The Professional could also be expanded up to two 3.5in disc drives and could also be input from two further 5.25in drives.

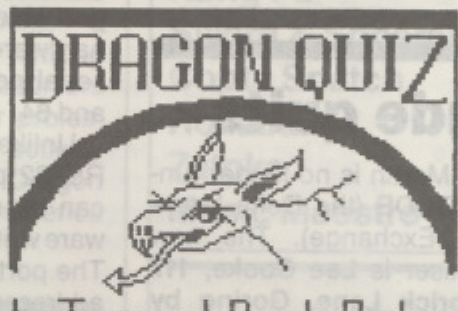
If anyone knows their whereabouts, please MBX me!

Roland Hewson
1 Swallow Gardens
Hatfield
Herts.
AL10 8QR

I have unearthed the original press report on the Professional from DU July 1984 and quote briefly ... "Project Veta really sees GEC Dragon hitting the big time with an expected retail price in the region of £2,500 to £3000 ... full production of the system is expected to go ahead in July with pre-production models already out." ... but the Editorial of that same issue, probably written very shortly before the magazine went to press, announces that the receivers have been called in on Dragon Data, by then already partly owned by GEC. "If GEC do take over Dragon Data, it is thought that they will concentrate on the Dragon Professional" says the August issue. By September, Eurohard had appeared on the horizon, and by October, production had moved to Carceres in Spain. By December, GEC had virtually washed its hands of the Dragon. Various upgrades were talked about subsequently, but the Dragon Professional never saw the light of day.

I have my doubts, but it is just possible that the fact that Dragon User didn't see a working copy of the Dragon Professional was deliberate caution on behalf of DD — there's an old saying that the best way to kill a bad product is to have a good advertising campaign. Revealing an unfinished machine to the minute scrutiny of the Dragon world would not have been a good move. Mind you, having it blow up in front of the PCW technical team wasn't exactly a PR coup, either ...

Crossword



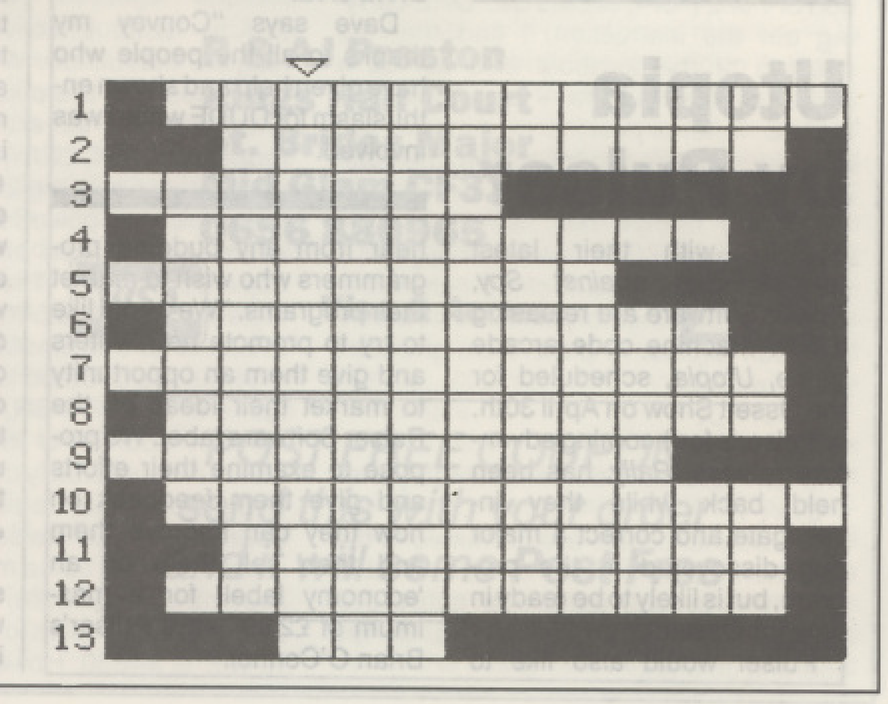
All this month's answers are names of Dragon software. When the crossword is complete, the column marked with an arrow will spell out a phrase.

The sixth Dragon Crossword is with us in time for Easter, if only just. Where crossword four is concerned, of course, there's no egg on the face of G. Wright from Dunblane, who doesn't say what he wants, but wishes me a happy holiday in retrospace (what a nice way of saying your Dragon's late) and J. Smith of Twyford who is similarly unspecific, but he has his answer right, so a winner he is.

There will be a couple of free tapes from the Editor's Magic Bottomless Box for the first correct entries to reach us each month. You can even try telling us which tapes you'd like in an ideal world. It all depends on what we can find.

And you don't have to cut up your *Dragon User*, either — entries can be written out on a photostat or a plain piece of paper, as long as we can read them.

1. Would he destroy a soapy circle? (6,6)
2. Must be a depressive one who works underground (5,5)
3. Viking in a frenzy (7)
4. A coward's race perhaps? (7,3)
5. A bell-ringer who got the hump! (9)
6. This was processed using a letter wire (10)
7. You will require a complete blackout to get this! (5,7)
8. Duo in danger from atack on their prison! (7,4)
9. Work and hag together, makes a fiery bird (6,4)
10. Bigger than a rock-fall (7,5)
11. Old age town caught up in the forest (10)
12. Take a plane after dark (5,6)
13. Easy if one is confused (3)



News desk

If you have any new products for the Dragon — software or hardware — ring the News Desk on 01-437 4343

Orange flowers

A NEW software house, Orange Software, has started trading from Abergavenny in Wales. Their launch list, suitably printed on bright orange paper, includes the following software, some of it old, some of it new:

Beanstalker on tape or disc for £3.99, along with a tape version only of *Beanpatch* for £1.00. Disc conversions of existing *Beanstalker* cassettes cost £1.00 — original inlay cards must be shown.

New games *Supernova*, a shooting game set in deep space, and *Matchmaker*, for younger users, four can play, for £2.99 each on tape or disc.

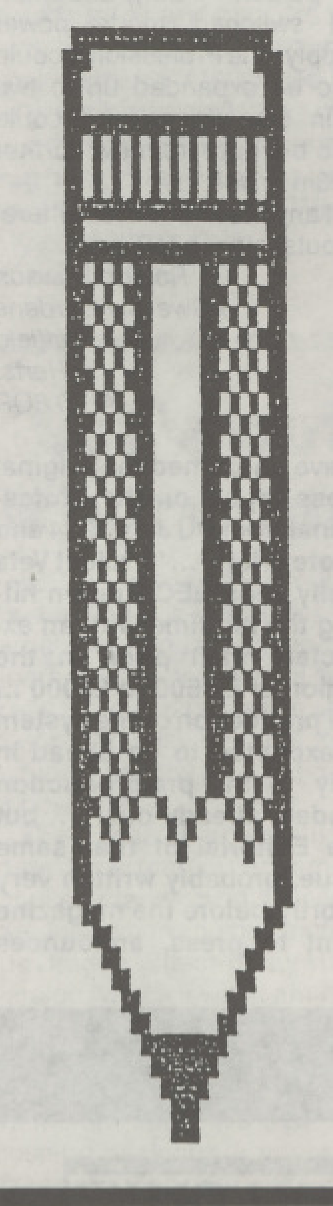
A new version of *Crazy Foota* for the Dragon and Tandy TRS80 32K.

New utilities *Sprite Designer* (tape or disc) for use with Basic or machine code programs and *Easel* (disc only) for use with Basic 42 by Harris Software are £4.99 each.

New utility *Text Screen Designer* for designing loading screens, £2.99 on tape or disc, and *Orange Boot*, a boot routine for many DOSs, disc only, £1.99.

They also have a list of several scheduled and recent adventure releases. Orange are looking for new Dragon/Tandy software.

For fact sheets, price lists and more information, send an SAE to Orange Software, The Garth, Star Road, Nant-Y-Derry, Abergavenney, Gwent NP7 8HF.



Utopia by Pulser

ALONG with their latest release *Spy against Spy*, Pulser Software are releasing a new machine code arcade game, *Utopia*, scheduled for the Ossett Show on April 30th.

Pulser's forthcoming adventure release, *Rally*, has been held back while they investigate and correct a major bug discovered in the program, but is likely to be ready in time for Ossett.

Pulser would also like to hear from any budding programmers who wish to market their programs. "We would like to try to promote new writers and give them an opportunity to market their ideas on the Pulser Software label. We propose to examine their efforts and give them feedback on how they can improve them and then sell them on an 'economy label' for a maximum of £2.99" says Pulser's Brian O'Connor.

Dude quits

Dave Martin is no longer running DUDE (the Delta Users Delta Exchange). The new organiser is **Lee Cooke, 117 Lumbrick Lane, Goring by Sea, Worthing, W. Sussex BN12 6AQ.**

Dave says "Convey my thanks to all the people who have given help and shown enthusiasm for DUDE while I was involved."

Just off the ground

REPORTS from the Cardiff Airport Show in February say that, although less than 100 people turned up at the door, the show itself covered its costs comfortably, and that attending retailers were well satisfied with their sales there.

Said organiser Helen Penn "We were running the Show on a co-operative basis, so that everybody who took part shared in the gate money. In fact, I have just mailed off an extra £15 to those people. Overheads were extremely low, so that we could take advantage of the site without having to draw a massive crowd." The remote site and a hitch in the advertising campaign probably conspired to keep away a few people who would otherwise have attended, but overall the Show was successful, and the Penns are now looking forward to the next show at the much more popular Ossett site on Saturday 30th April.

The contrast between the Cardiff Show and the sixth 6809 Show in London in December was pointed up by Robbie Preston of R & AJ Preston: "We sold plenty of games and did OK, but John and Helen Penn lost a certain amount of money on the show, because the overheads there were extraordinary." Finding an inexpensive site in central London has so far proved fruitless.

This experience must point in the direction of smaller, regional shows for the Dragon in future. The 1987 Ossett show was acclaimed by everyone who attended as a great success both financially and socially.

The Cardiff show could turn out to be a valuable pointer to keeping Dragon shows viable in a time of falling support for the Dragon.

Comms port for all

JIM Fuller, G4WPI has designed, tested and built an RS232 port which is both software and hardware compatible with the serial port on the Dragon 32 and 64.

Unlike many Dragon 32 RS232 ports, the new model can be used by comms software written for the Dragon 64. The port can be accessed via addresses &HFF04-&HFF07. The only two conditions affecting program compatibility are that the software must be able to run with only 32K of RAM, and must not use any calls to routines that are available only in the Dragon 64 rom.

This upgrade is contained on a small printed circuit board which fits inside the Dragon's case. The link with the outside world is via a 7-pin DIN socket on the left-hand side of the case, whose connections are configured to march those of the Dragon 64. Because the unit is permanently installed, the expansion port is free for use by a DOS cartridge.

The upgrade is available for self-fitting for £30, complete with fitting and programming instructions. Careful soldering with a fine iron is all that is required.

Alternatively, a soldering service for £8.50 is available from Chris Foster at 2 The Row, Berwick-St-James, Nr. Salisbury, Wilts SP3 4TP. Tel. 0722 790530.

These RS232 boards have already been installed in Dragon 32s by radio amateurs wanting to use radio-related serial hardware. The interface can be used to drive a serial printer, using a machine code routine and the ram hook at $167. A source listing for this is included.

For more information or a leaflet, send an SAE to Jim Fuller G4WPI, 42 Kitchener Road, Amesbury, Nr. Salisbury, Wilts SP4 7AD. Tel. 0722 790530.

No Chera

John Foster has written say that his projected software house Chera Designs (*Adventure Trail*, December 1987) will not now be going ahead 'owing to the smallness of the market'.

Quickbeam comes up against its final hitch

DAVE Hitchman's original software company, Quickbeam Software, has left the software business, selling its stock to Harry Massey at Computape.

Computape will now be sole suppliers of Quickbeam products.

Dave told *Dragon User* that pressure of work in his career had meant that he was no longer able to give customers the service they needed. "My customers will know that something's up, because I have quite a few letters that I haven't had time to answer yet" he said. "I'm travelling a lot more now, and I sold out to Harry because I reckoned that he could give people the better support than I could under the circumstances."

Thanking DU for its five-star review of *Indoor Football* in the- April issue, Dave said ruefully that it came just too late for him to benefit from the expected sales boost, "but I sold quite a few copies before the review, and it'll give Harry something to kick off on."

He stressed that the main reason he had chosen Computape to take over Quickbeam's list was that "I think they'll be around for a long time, and it was part of the agreement that they would support my previous customers, as well as just selling off the current stock."

Dragon User asked Dave if the criticisms that his prices were too high, made by vociferous sections of the Dragon following from time to time, had played any part in his quitting. He replied that it certainly had not.

"Someone rang me up from one of the fanzines and put that to me. I told him, if you think that you can publish original software for less, you go and do it."

"I know that the card wraps were cheap and nasty, but that is what you have to do to keep prices down. In their heyday, Microdeal were selling games for at least £8, and they weren't developing most of them. I bought in software because I wanted it, and I had to pay for it myself, from scratch. One of my programmers was offered £25,000 to go and work somewhere else, but he stayed to finish his work for me. How can we compete with that?"

"I didn't run Quickbeam to make a profit — I couldn't. I had to look at it as a mangement and marketing excercise. But my career has taken over my hobby. It's a pity. I would like to thank everyone who has supported us, and say I'm sorry to be leaving."

Other suppliers are finding it increasingly difficult to place original, pro-quality software on a competitive basis. Only recently, Pam D'Arcy said that her work on *Formula One* would be unlikely to cover costs, even if the game was popular.

Other software is being published by authors, or with no development advance to authors, in order to keep prices as low as possible. This is, of course, only possible where the author is running another, full-time, career.

Pundits have been spreading the rumour that Quickbeam would sell out for at least six months to date, but Dave Hitchman stoutly denies any plans to sell out before his career change intervened. This lends force to the feeling among Dragon professionals that spreading doom, a pastime favoured among some Dragon observers without a financial or professional stake, is actively detrimental to the Dragon market destroying confidence among Dragon users.

SOS-9 — alert over

Malcolm Cowan of the OS-9 Users Group (see April issue Stop Press) has written to say that Martin Vernon has now been in touch. He did not make clear whether the running of the group was back to normal, but we suggest that members and prospective members allow longer than usual to clear queries.

For those who wondered — no, it wasn't an April fool. They really did lose him.

Pamcodes

Part six of Pam D'Arcy's introduction to machine code

TO kick off with last month's hangovers:

(1) The program error in Yellow Blob is that the text screen is 512 ($200) bytes long and only 511 ($1FF) bytes are cleared

(2) The single instruction to use the 'clear text screen' ROM routine (JSR $BA77) introduced in December's article could be used to replace the instructions that initialise registers for and the actual 'CLEAR' loop instructions.

Auto increment and decrement

These are useful instructions when working through consecutive memory locations as they combine the functions of two separately available instructions, saving memory and execution time. Either of the four indexable registers, X, Y, S, U, may be used for the auto increment and auto decrement modes. Following the register in the operand column with + and ++ respectively causes the instruction to be carried out and 1 or 2 respectively to be added to that register. This is auto increment, sometimes also referred to as post increment.

In **listing 13** (last issue)(*Spot the editorial error. Let's see if we can squeeze it in again.*)

STA ,X+

thus copies the contents of register A at that time to the memory address contained in register X at that time; 1 is then added to the memory address contained in register X. It therefore gives the same result as

STA ,X
LEAX 1,X

but saves a line of source and object code and its execution time is faster than the pair of instructions. The format for auto incrementing by 2 is

STA ,X++

The auto decrement may also be referred to as pre-decrement because the subtraction of 1 or 2 (- or--) occurs BEFORE carrying out the instruction. I find that the position of the symbols also act as a memory jogger as the + sign follow the register=after or post increment, whereas the minus sign(s) occur between the comma and register.

STA ,-X

subtracts 1 from the contents of register X then copies the contents of register A to the contents=new memory address now contained in register X. It therefore gives the same result as

LEAX -1,X
STA ,X

The format for auto decrementing by 2 is

STA ,--X

No additional offsets are permitted in the auto increment/decrement modes; that is, no register or value (other than null) is permitted to precede the comma in the operand column.

In listing 13, register X is being used to contain the memory address of the text screen position that is currently being set to $60. After copying the $60 to that position, the auto increment adjusts the memory address to point to the next text screen position.

Useful application

A useful application of auto increment is when one wants to print strings of data. I have found this series hard work as I feel in a total vacuum. I am in the seventh article and there have been just two items of feedback from readers — one at the London Dragon Show requesting machine code routines for the likes of GET and PUT and the second from Denis O'Mulloy who sent me his version of December's 'print your name' workout. This is **listing 14**. It is obvious that he knows more machine code than an absolute beginner as I was not even intending a loop to be used let alone position independent code using an LEA instruction and use of auto increment mode to boot! However, I shall use it and build on it as he does use auto increment mode for displaying a string of data on the text screen. Let us consider the use of the LEA instruction first.

Actual memory addresses

I referred above to feeling like writing in a vacuum. Writing position independent (or relocatable) code can also be a little like that. One perhaps has several prompts or messages that one may need to ouptut to the text screen at an appropriate time yet, as the code may be loaded anywhere in memory how can one determine the actual address of a message for displaying on the text screen?

LEA — Load Effective Address is the magical instruction. We have previously only used it in the context of performing arithmetic on either of the indexable registers, LEAX, LEAY, LEAS, LEAU. The use that we are going to put it to now is just about the most powerful function of the instruction set when creating positon independent code. The same indexable registers (LEAX, LEAY, LEAS, LEAU) are available for this function.

When the operand is in the form of the name of a label, PCR, the actual current memory address of the first byte at that label name is actually computed for us and is placed in that register. Thus, to take **listing 14**, if the saved machine code is reloaded at the same memory address as assembled at ($4E20), the actual memory address of the start of the data at label NAME is $4E30. However, if the code is

Listing 13

```
5001                * LISTING 13
5001                *
5001                * YBLOB13 (FILENAME)
5001                *
5001                * THE YELLOW BLOB - PAGE 56
5001                * FROM "DRAGON MACHINE CODE"
5001                * BY JONES & COWSILL (SHIVA)
5001                *
5001                * CONVERTED TO BE RELOCATABLE
5001                * (LISTING 11+LISTING12)
5001                *
5001                * USING DSKDREAM ASSEMBLER
5001                * AFTER CLEAR200,&H5000
5001
5001 160004                LBRA   GO
5004
5004                WORK0  RMB    1      ;FOR 6400
5005                WORK1  RMB    1      ;    6401
5006                WORK2  RMB    1      ;    6402
5007                WORK3  RMB    1      ;    6403
5008
5008 8E0400         GO     LDX    #$0400
500B AF8CF6                STX    WORK0,PCR
500E 10BE01FF              LDY    #$01FF
5012 8660                  LDA    #$60
5014 6F8CEF                CLR    WORK2,PCR
5017 6F8CED                CLR    WORK3,PCR
501A A780           CLEAR  STA    ,X+
501C 313F                  LEAY   -1,Y
501E 26FA                  BNE    CLEAR
5020 869F                  LDA    #$9F
5022 AE8CDF                LDX    WORK0,PCR
5025 A784                  STA    ,X
5027 BD8006         KBD    JSR    $8006
502A 27FB                  BEQ    KBD
502C 815E                  CMPA   #$5E
502E 2604                  BNE    DOWN
5030 8D21                  BSR    UP
5032 20F3                  BRA    KBD
5034 810A           DOWN   CMPA   #$0A
5036 2604                  BNE    LEFT
5038 8D2F                  BSR    MDWN
503A 20EB                  BRA    KBD
503C 8108           LEFT   CMPA   #$08
503E 2604                  BNE    RIGHT
5040 8D3D                  BSR    MLFT
5042 20E3                  BRA    KBD
5044 8109           RIGHT  CMPA   #$09
5046 2604                  BNE    BREAK
5048 8D4B                  BSR    MRGHT
504A 20D8                  BRA    KBD
504C 8103           BREAK  CMPA   #$03
504E 2702                  BEQ    END
5050 20D5                  BRA    KBD
5052 39             END    RTS
5053
5053                * AMENDED LISTING 12
5053
5053 8600           UP     LDA    #$00
5055 A18CAF                CMPA   WORK3,PCR
5058 270E                  BEQ    ENDUP
505A 8660                  LDA    #$60
505C A784                  STA    ,X
505E 6A8CA6                DEC    WORK3,PCR
5061 3088E0                LEAX   -32,X
5064 869F                  LDA    #$9F
5066 A784                  STA    ,X
5068 39             ENDUP  RTS
5069
5069 860F           MDWN   LDA    #15
506B A18C99                CMPA   WORK3,PCR
506E 270E                  BEQ    ENDDWN
5070 8660                  LDA    #$60
5072 A784                  STA    ,X
5074 6C8C90                INC    WORK3,PCR
5077 308820                LEAX   32,X
507A 869F                  LDA    #$9F
507C A784                  STA    ,X
507E 39             ENDDWN RTS
507F
507F 8600           MLFT   LDA    #$00
5081 A18C82                CMPA   WORK2,PCR
5084 270E                  BEQ    ENDLFT
5086 8660                  LDA    #$60
5088 A784                  STA    ,X
508A 6A8DFF78              DEC    WORK2,PCR
508E 301F                  LEAX   -1,X
5090 869F                  LDA    #$9F
5092 A784                  STA    ,X
5094 39             ENDLFT RTS
5095
5095 861F           MRGHT  LDA    #$1F
5097 A18DFF6B              CMPA   WORK2,PCR
509B 270E                  BEQ    ENDRGT
509D 8660                  LDA    #$60
509F A784                  STA    ,X
50A1 6C8DFF61              INC    WORK2,PCR
50A5 3001                  LEAX   1,X
50A7 869F                  LDA    #$9F
50A9 A784                  STA    ,X
50AB 39             ENDRGT RTS
50AC
50AC
```

```
4E21                  * LISTING 14
4E21                  *
4E21                  * DENIS (FILENAME)
4E21                  *
4E21                  * DENIS O'MULLOY'S PRINT NAME
4E21                  * ROUTINE USING DREAM ASSEMBLER
4E21                  * AFTER CLEAR200,20000
4E21
4E20  4E20                  ORG   20000
4E20 BDBA77                 JSR   $BA77
4E23 318C0A                 LEAY  NAME,PCR
4E26 A6A0       LOOP        LDA   ,Y+
4E28 BD800C                 JSR   $800C
4E2B 8159                   CMPA  #'Y
4E2D 26F7                   BNE   LOOP
4E2F 39                     RTS
4E30 44454E4953 NAME        FCC   /DENIS O'MULLOY/
4E3E
```

```
5001                  * LISTING 15
5001                  *
5001                  * STRINGS (FILENAME)
5001                  *
5001                  * PRINT A STRING ROUTINE
5001                  * USING DSKDREAM ASSEMBLER
5001                  * AFTER CLEAR200,&H5000
5001
5001
5001 BDBA77                 JSR   $BA77
5004 308C19                 LEAX  NAME,PCR
5007 8D0D                   BSR   PRINT
5009 308C6C                 LEAX  ,NL,PCR
```

Listing 15 continued

```
500C BD0B                   BSR   PRINT
500E 308C1E                 LEAX  INFO,PCR
5011 8D03                   BSR   PRINT
5013 8D01                   BSR   PRINT
5015 39                     RTS
5016
5016                  * PRINT SUBROUTINE
5016                  * ENTRY: X=MEMORY ADDRESS OF
5016                  *         FIRST BYTE IN STRING
5016                  * EXIT : NULL BYTE ENCOUNTERED
5016                  *        IN STRING
5016
5016 A680       PRINT       LDA   ,X+
5018 2705                   BEQ   END
501A BD800C                 JSR   $800C
501D 20F7                   BRA   PRINT
501F 39         END         RTS
5020
5020 44454E4953 NAME        FCC   /DENIS O'MULLOY/
502E 00                     FCB   0
502F 44454D4F4E INFO        FCC   /DEMONSTRATION OF/
503F 4F4E45204D             FCC   /ONE METHOD FOR/
504D 0D                     FCB   13
504E 5052494E54             FCC   /PRINTING STRINGS/
505E 5553494E47             FCC   /USING A COMMON/
506C 0D                     FCB   13
506D 535542524F             FCC   /SUBROUTINE/
5077 00                     FCB   0
5078 0D00       NL          FCB   13,0
507A
507A
```

reloaded with an offset of &H1000 (=starting at memory address $5E20), the actual memory address of the start of the data at lable NAME would be $5E30. The magical LEAY NAME, PCR instruction takes care of this and will put into register Y the actual memory address of the start of the NAME date appertaining to the current load position of the code regardless of the memory address that the code was assembled at.

CCR FLAGS: LEAS, LEAU: no effect, LEAX, LEAY: zero flag

More Dream notes

You may be confused by a couple of new items that crop up in **listing 14**. Denis is obviously looking for the letter Y to terminate his loop. The apostrophe in the #'Y operand is Dream's method of indicating that the following (immediate) data is in the form of a normal printable ascii character rather than a decimal or hex number or value in any other format. If your assembler does not allow printable characters to be represented in the operand field, its ascii value in hex or decimal (#$59 or #89 respectivley) will be needed. The FCC line is another of what are known as assembler directives. It is Dream's method of allowing the programmer to define fixed (=preset) values in printable ascii character format. The / characters are one of Dream's means of indicating the start and end of a string of a character string; other assemblers may use different techniques and, in case one wishes to use the separator character itself within the data, often alternative separator symbols are available within the assembler. The string data alone between the separator characters are generated as object code, a byte per character. Program instructions generate between one and five bytes of object code in memory. Dream is among the types of assembler that commonly generates all the necessary bytes of object code (see the jump of 14 bytes to the next memory address used) but limits the printout of long source lines of preset data to a manimum of the first five bytes that it would need to print out with normal instructions.

When you get to **listing 15**, you will find FCB as well as FCC. These two directives are actually interchangeable in Dream but basically, FCC stands for defining Characters whereas FCB defines a Byte value. The comma separator allows more than one (consecutive) byte of data to be defined on a single line. This is an area where other assemblers are very likely to use different directives to define their preset data bytes (and different reserve memory bytes directives from Dream's RMB).

Print strings subroutine

If one has several messages that will need to be displayed on the text screen at an appropriate time in the program, one could repeat **listing 14**'s type of code for each message. However, what if the last character of the message appears elsewhere in the text, such as DENYS O'MULLOY? There are a number of ways of dealing with displaying text strings. One could load the length of the message into, say, register B, decrementing the count (DECB) after each character has been output until zero (null) is reached. The potential problem with loading the length as an immediate operand of a program instruction is that if one amends the content of the message, one has to remember to amend the program accordingly, too. An improvement could be to precede the message data itself with a byte containing the length thus:

```
NAME    FCC     14,/DENIS O'MULLOY/

        LEAX    NAME,PCR
        LDB     ,X+         ;GET COUNT
LOOP    LDA     ,X+
        JSR     $800C
        DECB
        BNE     LOOP

        RTS     COUNT EXHAUSTED
```

which is a more immediate reminder to also maintain the length when amending message content.

Rather than have to include the nature of a full display loop for every message, it cuts down on the number of instructions=opportunities for error=length of program=code to be tested if the common parts of code such as of the display code is made into a subroutine within our program. Subroutines are usually pieces of code that carry out specific or often repetitive task. Messages that we want displayed (1) won't necessarily always be the same length (2) will be located at different memory addresses within our program (3) won't necessarily be displayed at the same screen start position.

We need to pass such information, then, to the subroutine as parameters — or values set up in registers or memory — to enable the subroutine to perform its tasks. In this instance, the memory address of the message to be displayed and be passed across in a selected register. We are using the normal text display ROM call $800C that uses the 'text screen pointer' at memory addresses $88,$89 as its curent cursor position so we can adjust that location should we wish to deviate from the norm with message positioning. That leaves coping with variable length messages.

Yet another method for dealing with the output of messsages of varying length that would not need a length byte to be maintained nor decremented is to use a special terminating character. As a character from the string is copied into register A in readiness for display, it could be checked for 'end of string' reached, rather as Denis

checks for the letter Y (but after display) in his routine. If a teminator such a s null is chose, a CoMPare instruction is not needed as the very action of copying the next character of the string into register A (LDA ,X+) sets the zero flag of the CCR if a null is encountered enabling null to be instantly tested for and an exit to be made from the subroutine back to the calling program.

I will now leave you to fathom out **listing 15**. Decimal 13 ($0D) is the ascii code for carriage return =go to the start of the next screen line (also-enter key when obtaining keypresses). One could build anything that one wanted to in a common subroutine. For instance, one could automatically output a carriage return before RTS when the null was encountered. However, there may be times when it is very useful not to always do so, such as prompting for input on the same line. One could always precede the terminating null with a 13 where one specifically wanted to set the display ready at the start of the next line. Note in **listing 15** where BSR PRINT is repeated without an intervening LEAX. This is illustrating where one can take positive advantage of register X being auto incremented and left in that final state by the subroutine (which is in this instance, printing to the start of the next message) for displaying a number of consecutively held messages.

Should you try **listing 15**, you will find that I am not infallible even when writing a small piece of code quickly (or , come to that, any other size or any other time!). No problems with it working — just silly slips in the message content for you to put right!

That's another space allocation done for, I'm afraid, so positive and negative numbers will be dealt with next month — positive!

Dragonsoft

New software for review should be sent to *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

Amenophis the simple

Title: *Pyradventure*
Price: £3.00 plus 50p p&p
Supplier: Dragonfire Services

ALTHOUGH at first this program may sound like an adventure about an arsonist it is in fact one set in the pyramids of Egypt. Not an idea that is new to adventures but one which undoubtedly offers a lot of scope to the writer ever striving for inspiration.

The idea may not be new but its producers are new and expanding: Dragonfire Services, set deep in loyalist Dragon country of Wales, where the machine was raised and it seems is now being nurtured in its old age. Dragonfire have however been running a Dragon magazine for a couple of years and also have other software such as *Underbeings of Croth* in their range which if my memory and a quick flick through some back issues of DU serves me right, used to be marketed by a firm called Maridan.

Your task in the game is to explore the pyramid in which the tomb of Amenophis III, the father of Tutankhamun is located. Once in the tomb you must obtain his gold and silver and the death mask of Amenophis himself.

After a graphics screen loader and scene setting instructions you find yourself in a small shop where there are various items to buy, but alas you don't have any money (at least one aspect is true to life anyway). Undeterred by having none of your allowable five items on your inventory list you can venture outside and make your way to the pyramid, although to get in there you will need light. First task therefore is the original adventure guideline of examining everything in sight.

Having solved the first problem you've more or less free to wander at will, although there is a collapsing floor if you're not careful and the vicious cobra to get past. There is plenty of time to think of how to solve your problem though as the game is not played in real time contrary to the popular trend.

The usual verb-noun situation applies with directions shortened to one letter, ie 'N', although 'go' has to be prefixed to the direction. Vocabulary is limited but if you can't find the right words for the task you're either doing something unnecessary or are suffering from a serious case of verbal delinquency.

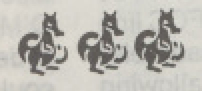
Trimmings to the game include the freeze/unfreeze command which allows you to save a position in the computers memory and return to it if necessary, a useful help when trying to find out which room to blast to smithereens with the dynamite. There's also a save/load command to use if you can't bear to tear yourself away but drastic events interfere like the need to make a living, or a herd of stampeding elephants about to plough through your front door.

I like this game, but, and this is a but I don't often use in reviews, it's one that is too easy. Even at my leisurely pace of playing, I completed it quickly and although there's always the satisfaction of succeeding, I feel a touch more intrigue, even the stampeding elephants I just mentioned, would test the brain to a greater degree.

Drawing to a close though, this is a logical adventure where problems are solved by doing what you'd do in real like, not by obscure phrases and by chance. It may not be vast, doesn't have redefined text, but it is interesting to play (and also on the cassette inlay it says "real speech", but despite completing the game my television hasn't talked yet — apologies to Dragonfire if it's just my Dragon that's mute).

This certainly leaves me looking forward to seeing more from this firm, and they promise more titles. I'm only sorry I can't give this more than three Dragons. But their description of it being a 'humdinger' is a bit of an exaggeration.

Philip Stott



Go boldly in reverse

Program: *Space Trek* and *Reversi* on one cassette
Supplier: Preston
Price: £2.99

I must confess to some considerable surprise that anybody should wish to re-release two games that never made it into the charts originally. These were both put out by Trojan in 1982, both written in Basic and no attempt appears to have been made to update them. *Space Trek* is still in the original form, still contains the unforgettable double 'n' in enginneer (*Once I couldn't spel enginneer — now I are one.*) in the instructions, and unfortunately is as bad as ever. The screen display is too small, totally confusing and the temptation to press the Break key is almost unbearable. The game can be played with a jopystick or keyboard and adds the comment in the instructions that "This may seem a little difficult at first, but it will only take a few minutes to get used to it".

The object is to destroy all the Trojans in the galaxy in a given time with the aid of a battle computer which will automatically fire the ship's phaser banks when you place the ship in set positions in relation to the Trojans, ie from the four conrners and the four sides. There is a range scanner to enable you to see the content of any space quadrant up to seven quadrants until the scanner is damaged when its range decreases rapidly. In order to proceed through the galaxy and reach the objective you have, of course, both impulse drive and warp drive, energy shields and 'di-lithium crystals' to take to the star bases, and finally a considerable assistance, to have a pencil and paper to note the positions of the star bases, planets, etc.

Unfortunately, for this version of *Space Trek* both Salamander and Wintersoft

Continued on next page

Dragonsoft

New software for review should be sent to *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

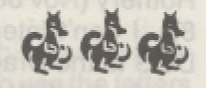
Continued from previous page

produced versions called *Dragon Trek*, Salamander with a twelve page flight manual, and Wintersoft one much nearer to the original, both of them superior to this But all three are infinitely forgettable when 'boldly going' in front of an episode of the TV series.

Reversi or *Othello* as it is sometimes known is perhaps worth a re-issue if coupled with a chess or draughts as a twosome, but it appears to me a peculiar choice to partner with *Space Trek*. However, the game is very well displayed, black is blue, but the graphics are splendid. The computer's game is a little slow even with the speed poke and it seems to spend a lot of time 'thinking'. There are four levels of skill the first of which is rather easy as I beat the computer by a wide margin, armed with a false sense of complacency I immediately jumped to the hardest level where my ego was dealt a severe blow! This program is superior to the Oasis version which only had two skill levels and I found it quite absorbing. My wife usually beats me at the board game itself, so I have to lose gracefully; with the coomputer I can turn it off and not tell anyone!

As a whole for those who do not have other versions Preston are to be congratulated for producing a package cheaply, but I feel their later disc package of earlier games are better value for money, and perhaps a better choice. Only one dragon for *Space Trek*, three for *Reversi* and four for Preston themselves for offering the re-runs for those who missed the originals. I would suggest that they improve the presentation. Cheapness is no excuse for obvious spelling mistakes!

R L N Hewson



Another foot onwards . . .

Name: *Crazy Foota 2*
Supplier: Computape
Price: £2.99. Dragon 32 or 64, one or two joysticks (switched or pot). One or two players. Cassette or disc

Crazy Foota 3 now available from author Grahame Smith at Orange Software, £2.99.

YET another football game (we're spoiled for choice), this time coming from Computape, continuing their Crazy Foota series.

Crazy Foota 2 is described as being greatly improved on the old version. In many ways it is leagues above *Crazy Foota*, but it too has its faults.

The game takes surprisingly little time to load. After a well drawn title screen comes a small burst of music. Although not of the same standards of *Superkid*, *Rola-Ball* etc. it's much improved on the few beeps we get in C.F 1. First come the instructions (as usual I didn't read them). Next you are shown a list of options. Among them you can select a single player game against the computer, you can display the high score table or you can start the game.

However there's another option. Pressing 'O' displays the 'other options'. This is where *Crazy Foota 2* comes into its own. You can select/change the speed of your players, alter the time set for each game and select names for both the red and yellow teams. Red and yellow in this case because this game's in colour.

Having selected all you want, you can then begin the game properly.

You can either choose to play an opponent or the computer. The main drawback with

Continued on page 11

The Tandy Disc

Eric Hall describes the unusual disc set-up of the Tandy CoCo

I HAVE been prompted to write this article, having read similar articles from D. Rothery (Nov 86) and Paul Dagleish (May 87). I don't intend to compare the Tandy DOS to the Dragon DOS. I have used the previous articles as a basis. It may also be of use to D.Mcquade of New Zealand who had a problem with 35 and 40 track discs.

Tandy Color Computer DOS users are aware the Tandy disc is set up differently to most home computers because it has only 35 tracks. Since each track contains two granules or 4,608 bytes, one granule contains 2,304 bytes. There are 68 granules to a disc.

The 68 granules are numbered 0-67 for reference and are located as follows:

Track 0, sectors 1-9 granule 0
Track 0, sectors 10-18 granule 1
Track 14, sectors 1-9 granule 28
Track 17, sectors 1-18 Directory
Track 18, sectors 1-9 granule 34
Track 24, sectors 10-18 granule 47
Track 34, sectors 10-18 granule 67

The Tandy CoCo uses these granules to allocate space for disc files in 2,304 byte clusters. So, if a file contains 4,700 bytes, the CoCo allocates 3 granules (6,912 bytes) of Dics space for it. Each track on the CoCo also contains 18 sectors, numbered 1-18. Each sector holds 338 bytes, of which 256 bytes hold data. The remaining bytes are used in the systems controls:

Byte contents
0-55 Systems controls.
56-311 Data.
312-337 Systems controls.

I would like to mention, just for interest, the hexadecimal value of the system controls as listed in the Tandy disc manual (Page 58):

Byte	hexadecimal
0-7	00
8-10	F5
11	FE
12	Track number.
13	00
14	Sector number.
15	01
16-17	Cyclic redundancy check (CRC).
18-39	4E
40-51	00
52-54	F5
55	FB
312-313	Cyclic redundancy check (CRC).
314-337	4E

The disc directory track

The Tandy DOS directory is on track 17, it is here that the CoCo stores the file allocation table and up to 72 directory entries. The information is stored on sectors 2-11 as follows:

Sector contents
2 File allocation table or granule map.
3-11 Directory entries.
11-15 Holds system bytes.
16-31 Not used by present system and can be used as disc labels, as Mr. Rothery suggests in his article of November 86.

File allocation table

The file allocation table or granule map is the disc's way of knowing where to find the next item of data in a program or file. This information is stored in the first 68 bytes of sector 2, track 17. The 68 bytes of the sector relate directly to the 68 granules the CoCo uses to allocate space for data. These bytes will either contain a value of &HFF, &H00-43, or &HC0-C9:

FF The corresponding granule is free. It is not part of a disc file.

00-43 The corresponding granule is part of a disc file. If byte 16 has the value &H12 in it, the next granule in the file will be granule 18(decimal).

C0-C9 The corresponding granule is the last granule in the file. The value contained in bits 0-5 of this byte tell how many of the sectors in that granule are used up in the data file. (Bits 7 and 8 are both set.)

This is a typical granule map when dumped to the printer:

```
0000 C5 C2 03 00 C3 02 C2 C2 09 C1 C1 08 C2 0A C3 C3
0010 C7 0E 13 10 C4 12 17 14 C9 16 1B 18 C3 C3 1F C4
0020 21 C3 23 C3 C5 26 27 C5 C1 2A 2B C9 2D 2E 2F C6
0030 31 32 33 C7 C2 36 C3 C4 C8 3A 3B 3C C6 C3 C6 C0
0040 C7 C2 43 C1 00 00 00 00 00 00 00 00 00 00 00 00
```

Here I have shown the first 80 bytes of track 17 sector 2. All values after the first 68 bytes are not used and hence have the vlaue 00 im them.

One way of write protecting the Tandy disc is to use a short program in Basic or machine code that will find the &HFF value in the allocation table (unused granules) then change these values to &HC0. this has the effect of fooling the Disc into believeing it is full and return the error message DISC FULL.

```
Directory Entries, Format:

00 01 02 03 04 05 06 07:08 09 10: 11 : 12 :  13   : 14  15 :16-31:

<------Filename--------><--Ext--><File><Flg><-Len-><No.bytes><-nop->
```

```
000 4D 45 52 47 45 20 20 20 55 4C 54 00 00 0C 00 3B  MERGE...ULT.....
010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
020 4D 4C 20 41 44 44 52 20 42 41 53 00 00 24 00 4F  ML.ADDR.BAS.....
030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
040 4F 52 47 41 4E 20 20 20 53 43 20 00 00 3E 00 18  ORGAN...SC......
050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
060 50 4C 41 4E 52 4F 4F 4D 42 41 53 00 00 0B 00 C2  PLANROOMBAS.....
070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
080 53 2D 45 46 46 45 43 54 53 43 20 00 00 40 00 A3  S-EFFECTSC......
090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0A0 53 43 52 4E 44 55 4D 50 43 47 50 00 FF 42 00 4F  SCRNDUMPCGP.....
0B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0C0 53 43 52 50 52 54 20 20 42 49 4E 02 00 1C 00 3A  SCRPRT..BIN.....
0D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0E0 53 43 52 50 52 54 31 20 42 49 4E 02 00 1D 00 3A  SCRPRT1.BIN.....
0F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Byte contents

0-7 Filename, is left justified and blank-filled. If byte 0=0, the file has been deleted and the entry is made available again.

If the byte 0=&HFF, the entry and all following entries have not yet been used.

8-10 file name extension, left justified, blank filled and may be assigned for users reference.

11 file type
0=Basic program.
1=Basic data.
2=Machine-code program.
3=Text editor
12 ASCII flag
0=Binary format.
FF=ASCII format.
13 The number of the first granule in the file (0-67).
14-15 The number of bytes used in the sector of the file.
16-31 These bytes are not used.

Once again we have a dump, track 17, sector 6. Figure 1 shows a part of the directory from Tandy DOS. From this we can get the file data:

Bytes 0-7 gives us the filename. If you look at the values you will see these represent the ASCII code for the filenames listed to the right.

Bytes 8-10 gives us the extension used.
Byte 11 is the file type, 0=Basic, 1=Basic data, 2=machine code, 3=text editor source code.
Byte 12 The ASCII code flag 0=binary, FF=ASCII code.
Byte 13 The starting granule on the disc.
Byte 14 Reports the number of bytes used in the last sector of the file.

If we look closer at the files, for example, the filename 'SCRNDUMPCGP' The value of byte 15 we have is &H4F or 79(dec.). This indicates only 79 bytes were used on the last sector.

If we read right to left the value at Byte 13 is &H42,(66 dec.). Therefore the first granule used in this file is on track 34, Sector 1 (granule 66). Further left the next value is at Byte 12, &HFF, which shows this file was saved in ASCII format as the flag is set. Once the CoCo picks up the first location in a file it will then go to that granule position on the allocation map (track 17, sector 2), to pick up the next granule and os on until it finds a last granule marker. Then the CoCo knows the file is completely loaded.

Lastly a mention about the command 'DOS'. When DOS is typed in the Tandy disc goes to track 34 and checks for the letters 'OS'. If present it then copies the entire track into memory. If (S) is present then it will boot the system.

If anyone wishes to use this idea for their own programs remember to use the letters 'OS' at the start of your program. Also you must allow five bytes space in front of the routine for the Tandy DOS to use for system control bytes.

Tandy DOS system usually starts loading the first data block at granule 32 on track 16. It is a good idea to start Tandy 'DOS' booted routines at track 16 and follow in the system.

As this format shows, the directory files are stored differently to Dragon DOS. Also to be noted is that Tandy DOS has no file write protection facility in byte O. I have listed the details for the directory file below.

Dragonsoft

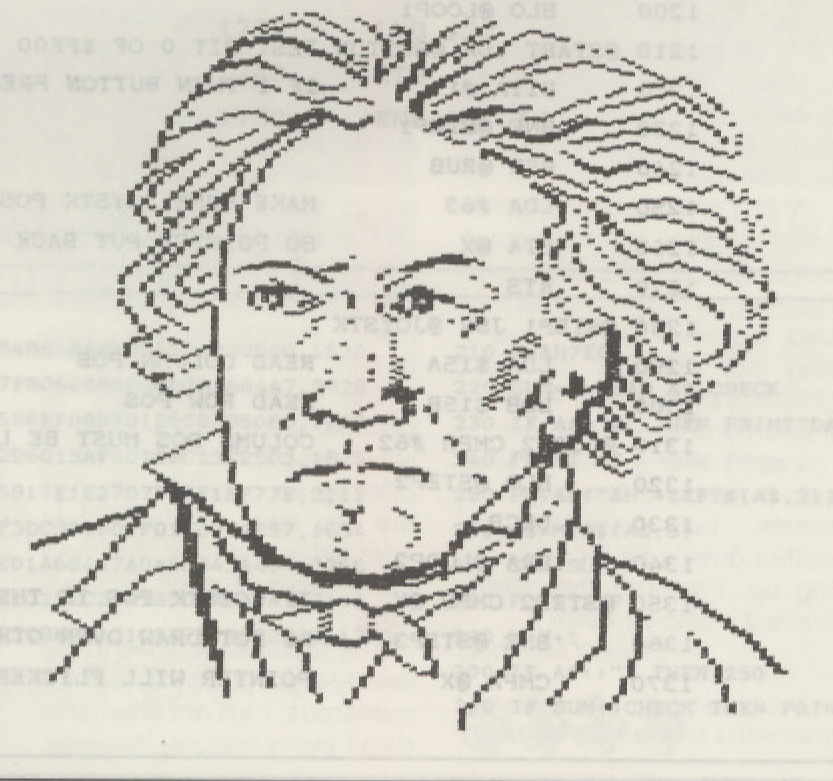
New software for review should be sent to *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

Every picture needs a thousand words

Program: *Picture Maker*
Supplier: John Penn
Price: £5.00

THE main disadvantage of using the higher resolution graphics modes is the comparatively large amount of data that needs to be input in order to produce quite modest results. Something as simple as a box viewed corner on would need a minimum of nine lines drawn on the display, as well as having to determine the exact screen locations for drawing the lines. What is needed is an on-screen graphics utility package to enable the creation of hi-res displays — with the facility to amend the display, and store it for recall at a later date.

Enter *Picture Maker* from John Penn Software. This utility allows the design of PMODE3 graphics using mainly the four cursor keys (or optional joystick) to draw the lines. Each line can be drawn in turn and altered until it is correct before being 'entered' (or cancelled if it is not satisfactory). The example shown here was one which I was able to produce in a couple of hours using the package — plus a suitable illustration as a guide. Now, I must admit to being totally unable to draw anything on paper and so found the 'try it and see' ability provided by *Picture Maker* particularly useful. Also, the availability of the GET and PUT commands to move whole areas of the screen display around was very important. In drawing the portrait I started with the eyes, and finding them too close together I was able to relocate them until they were correctly placed. (The image of Ludwig van Beethoven's eyes wandering around a video screen is the stuff of nightmares!) Once the display is complete it can then be filed to tape.



That was the good news — now for the bad. What would appear to be a very useful package is let down by inadequate documentation. The double-sided A4 sheet provided needs to be considerably expanded in order to explain more fully the functions available. I'm sure that there must be other functions available but, frustratingly, it was impossible to find what they were! For example, mention is made of three 'screens' — the 'view' screen, which is presumably the one producing the display — plus a 'user' screen, and an 'auto' screen. What these are and what they do is not made clear! Also, the section on the brush functions would benefit from considerable expansion.

Unfortunately, the screen dump program supplied with the package proved incompatible with the printer that I was using due to codes being required which were not recognised by the printer. This difficulty was overcome by retyping line 0 of the screen dump program provided, on to the beginning of a suitable dump program taken from the pages of *Dragon User*. (It was necessary to alter the PPOINT values to read 0 or 1 when doing this.)

In summary, given clearer documentation, this would appear to be a very comprehensive package but the fact that I was unable to use it to anywhere near its full potential makes an accurate assessment difficult. One dragon as it stands, but I'm sure another two with a revised instruction sheet.

Gordon Lee

Getting the point

Nigel Mason shoots an arrow at the Dragon

IF you are a little jealous of all those other computers that have a mouse controlled environment, but you have a potentiometer type joystick, then I offer a partial solution. (Could someone do a review of Harris's *KLIK* utility for *BASIC42* please?) **Listing one** in DASM format assembler gives a non-destructive pointer on PMODE4 screens, which returns control to Basic when the fire button is pressed (and certain other keys on the keyboard). A pointer and on-screen Icons give the programmer much better control over user input and is ideal for users who are unfamiliar with a keyboard. Alternatively type in **listing two** and then save it before running. The checksum should spot any errors. If all is fine, then save the code with

CSAVEM"POINTER",&H7E01,&H7EA2,&H7E20.

Also before EXECuting the code have at least four graphics pages reserved and a joystick plugged into the right-hand joystick port. EXEC &H7E20 to run the code. If you want to use other graphics pages then POKE &H7E65 with the hex value from the following table:

Start	page	1	2	3	4	5
POKE	value	06	0C	12	18	1E

Use the following functions to find the co-ordinates of the arrow point:

```
DEF FN X(X)=(PEEK(&H15A)
AND &HFE)*4
DEF FN Y(Y)=(PEEK(&H15B)+2*
(PEEK(&H15B)>61))*3
```

and

```
X=FN X(0)
Y=FN Y(0)
```

You must have four consecutive graphics pages reserved from the start page, otherwise the pointer will overwrite the Basic area, so PCLEAR enough space. To detect the joystick button being pressed, use:

```
PEEK(&HFF00) AND 1 = 0 BUTTON NOT
PRESSED
                   = 1 BUTTON PRESSED
```

Now, how to define your own pointer: the pointer is defined on an 8 column by 9 row grid with two bytes used to define each row. The first byte defines the pointer and the second the highlight which makes the pointer visible in black areas of the screen (ie the column positions that always remain unset). The numbers are poked alternately into consecutive addresses starting at &H7E01. As an example look at the pointer defined in the program.

```
7E01                 1040      PRT
7E01 00FC7884709C    1050 @ARROW FCB $00,$FC,$78,$84,$70,$9C
7E07 50A848B404EA    1060      FCB $50,$A8,$48,$B4,$04,$EA
7E0D 020500030000    1070      FCB $02,$05,$00,$03,$00,$00
7E13                 1080 @SCREEN RMB 9      BYTES FOR BACKGROUND
7E1C 0600            1090 @POS FDB $0600     CONTAINS ADDRESS OF POINTER
FF00                 1100 @BUTTON EQU $FF00  CONTAINS STATE OF FIRE BUTTON
BD52                 1110 @JOYSTK EQU $BD52  ROM ROUTINE TO READ JOYSTK PORTS
7E1E 00              1120 @X FCB 0           COLUMN POS
7E1F 00              1130 @Y FCB 0           ROW POS
7E20                 1140 @BEGIN EQU *
7E20 8D78            1150      BSR @LOAD1
7E22 A684            1160 @LOOP1 LDA ,X      INITIAL STORE OF BACKGROUND
7E24 A7A0            1170      STA ,Y+
7E26 3A              1180      ABX
7E27 108C7E1C        1190      CMPY #@SCREEN+9
7E2B 25F5            1200      BLO @LOOP1
7E2D B6FF00          1210 @START LDA @BUTTON TEST BIT 0 OF $FF00
7E30 8501            1220      BITA #1       IF 1 THEN BUTTON PRESSED
7E32 2608            1230      BNE @STEP1
7E34 8D56            1240      BSR @RUB
7E36 863F            1250      LDA #63       MAKE SURE JOYSTK POS HAS CHANGED
7E38 B77E1E          1260      STA @X        SO POINTER PUT BACK ON RE-ENTRY
7E3B 39              1270      RTS
7E3C BDBD52          1280 @STEP1 JSR @JOYSTK
7E3F B6015A          1290      LDA $15A      READ COLUMN POS
7E42 F6015B          1300      LDB $15B      READ ROW POS
7E45 C13E            1310 @LOOP2 CMPB #62    COLUMN POS MUST BE LESS THAN 62
7E47 2503            1320      BLO @STEP2
7E49 5A              1330      DECB
7E4A 20F9            1340      BRA @LOOP2
7E4C F17E1F          1350 @STEP2 CMPB @Y     IF JOYSTK POS IS THE SAME
7E4F 2605            1360      BNE @STEP3    DO NOT DRAW OVER OTHERWISE
7E51 B17E1E          1370      CMPA @X       POINTER WILL FLICKER
```

```
7E54 27D7        1380        BEQ @START
7E56 B77E1E      1390 @STEP3 STA @X          STORE COLUMN POS
7E59 F77E1F      1400        STB @Y          STORE ROW POS
7E5C 8D2E        1410        BSR @RUB
7E5E 8660        1420        LDA #96         CALCULATE POINTER ADDRESS
7E60 F67E1F      1430        LDB @Y          IN SCREEN MEMORY
7E63 3D          1440        MUL
7E64 C30600      1450        ADDD #$0600     START OF PAGE 1
7E67 1F01        1460        TFR D,X
7E69 F67E1E      1470        LDB @X
7E6C 57          1480        ASRB
7E6D 3A          1490        ABX
7E6E BF7E1C      1500        STX @POS        STORE ADDRESS OF POINTER
7E71 8D2A        1510        BSR @LOAD2
7E73 CE7E01      1520        LDU #@ARROW
7E76 A684        1530 @LOOP3 LDA ,X          STORE BACKGROUND AND
7E78 A7A0        1540        STA ,Y+         PUT POINTER ON SCREEN
7E7A A6C0        1550        LDA ,U+
7E7C 43          1560        COMA
7E7D A484        1570        ANDA ,X
7E7F AAC0        1580        ORA ,U+
7E81 A784        1590        STA ,X
7E83 3A          1600        ABX
7E84 108C7E1C    1610        CMPY #@SCREEN+9
7E88 25EC        1620        BLO @LOOP3
7E8A 20A1        1630        BRA @START
7E8C 8D0C        1640 @RUB   BSR @LOAD1      PUT BACKGROUND BACK
7E8E A6A0        1650 @LOOP4 LDA ,Y+
7E90 A784        1660        STA ,X
7E92 3A          1670        ABX
7E93 108C7E1C    1680        CMPY #@SCREEN+9
7E97 25F5        1690        BLO @LOOP4
7E99 39          1700        RTS
7E9A BE7E1C      1710 @LOAD1 LDX @POS        COMMON LOAD INSTRUCTIONS
7E9D 108E7E13    1720 @LOAD2 LDY #@SCREEN
7EA1 C620        1730        LDB #32
7EA3 39          1740        RTS
7EA4             1750        END @BEGIN
```

```
100 DATA 00FC7884709C50A848B404EA020500030000,1520
110 DATA F0F1F2F3F4F5F6F7F8060000008D78A684A7,2928
120 DATA A03A108C7E1C25F5B6FF00850126088D5686,1788
130 DATA 3FB77E1E39BDBD52B6015AF6015BC13E2503,1825
140 DATA 5A20F9F17E1F2605B17E1E27D7B77E1EF77E,2111
150 DATA 1F8D2E8660F67E1F3DC306001F01F67E1E57,1634
160 DATA 3ABF7E1C8D2ACE7E01A684A7A0A6C043A484,2265
170 DATA AAC0A7843A108C7E1C25EC20A18D0CA6A0A7,2141
180 DATA 843A108C7E1C25F539BE7E1C108E7E13C62039,1773
190 DATA @,0
200 CLEAR 200,&H7E00
210 X=&H7E01
220 SUM=0:READ A$,CHECK
230 IF A$="@" THEN PRINT"DATA CORRECT":END
240 PRINT A$
250 P=VAL("&H"+LEFT$(A$,2))
260 A$=MID$(A$,3)
270 SUM=SUM+P
280 POKE X,P
290 X=X+1
300 IF A$<>"" THEN 250
310 IF SUM<>CHECK THEN PRINT"ERROR" ELSE 220
```

```
100 PMODE 4,1:SCREEN 1,0:COLOR 0,5:PCLS 5
110 DEF FN X(X)=(PEEK(&H15A) AND &HFE)*4
120 DEF FN Y(Y)=(PEEK(&H15B)+2*(PEEK(&H15B)>61))*3
130 FOR N=1 TO 50
140 LINE-(RND(255),RND(191)),PSET
150 NEXT
160 EXEC &H7E20
170 PAINT(FN X(0),FN Y(0)),0,0
180 GOTO 160
```

Pointer	Highlight
00	FC
78	84
70	9C
50	A8
48	B4
04	EA
02	05
00	03
00	00

Dragonsoft

New software for review should be sent to *Dragon User*, 12-13 Little Newport Street, London WC2H 7PP.

From page 11

playing the computer is that if you play in Yellow you have to use the left joystick, whereas if you change to Red you've also got to change joystick.

As in C.F. 1 there are eleven players in each team, all down your side of the screen. You control the team in your colours and your opponent controls the other team.

The object of the game is not simply to score as many goals against your opponent within the set time limit. In this game you can choose to play 'first to five' or 'first to ten'.

The method of play is almost identical to the first version. To select a player you simply move a pointer up the screen, by pressing the button and moving the joystick up or down. Moving the joystick moves your selected player.

If you manage to pick up the ball you can run with it, shoot or pass. You can tackle another player simply by standing in his way. Alternatively you can tackle while on the run but the chance of success are said to be 50%.

Having got the ball, pressing 'fire' kicks the ball in the direction you are facing, however you cannot kick the ball horizontally.

This time the 'goals' have been enlarged much to my distress. With the old version you could score simply by kicking the ball either above or below the goaly. Well its more of a problem now, because the goals are larger, meaning more space between the 'keeper' and his surrounding players.

The problem of being able to walk straight past a player has gone (now you're almost certain to be tackled) and the problem of being able to walk straight into the goal has also gone. Now you can only score with a diagonal shot unless the ball deflects off another player and goes into the goal.

The graphics are not quite as good as in C.F 1. In particular the players. You should see the way they run! They seem to drag their legs across the ground. However that shouldn't prove a problem. The cheers are still the same, although this there's an alarm at the end of the game.

I don't like the way this game is laid out. If you're going to have a game where you control all the players then I'd opt for the way Quickbeam have chosen. The pitch is rather small on this game. It's better to spread the players out on a large pitch that scrolls along.

That said however, *Crazy Foota 2* does make up for its faults. There's a high score table, as in the first version.

Crazy Foota 2 is indeed greatly improved. It offers some very good features and provides hours of fun.

At first when I saw this game I was appalled at the graphics but once you settle down, you realise that they're not all that bad. So while *Crazy Foota 2* may not possess the best graphics and sound I've seen, it is fun and quite addictive.

I won't offer it five Dragons yet. It needs a few improvements, but providing the author does a bit more touching up, I'm sure that *Crazy Foota 3* would be worth it.

Donald Morrison



Point of destruction

Program: *Printer Control*
Supplier: McGowan Consultants
Price: From £19.95

WHERE would we be without word processors? "A lot better off in the pocket!" used to be my decidedly naive attitude towards the journalist's supposedly greatest companion. However, such an attitude was drastically overturned when, like pennies from heven, a revolutionary piece of software found its way into my delerious tape recorder.

This magical utility was none other than the dot matrix version of *Printer Control*. While not being new, *Printer Control* is one of the less publicised Dragon utilities, which is surprising considering the sheer wealth of content contained in it.

On loading, the first thing to greet you is a menu; one doesn't even have to get tangled up in the complexities of configuring, as the ever helpful McGowan are ready at hand, uniquely providing a tailor-made version to suit your printer.

The menu comprises ten options, allowing you to load/save/merge/print files, while offering an option to change the key rate, edit, save/load user defined strings, and view the amount of memory available.

The features outlined in the twenty-eight page manual are numerous; I will not attempt to detail every feature. However, the core of the program is prominently devided into two modes, text and graphics.

Based on a forty column screen, the text mode simply offers the basic word processing features. The editing facilities, while being simplistic, give you easy access to manipulation. Also offered is right hand justification, allowing files to be printed in a tidy block, string detection, various type styles (including inverse text) depending on the capabilities of your printer, and limited graphics from the vast array of graphics characters directly accessible. However, for more detailed graphics, a special graphics mode is available.

Endorsing that this is not 'merely a word processor', this second mode is picture mode, which, as the name implies, allows you to create your own pictures ready for printing. Virtually any form of graphics can be produced from the vast array. An additional option to create your own graphics characters is also included, for people whose needs are more specific. The only restriction on graphics is the size of the screen, which is this time based on 32 columns with 24 lines.Although this can be overcome to a certain extent by simply altering the size of the printout.

Restrictions aside, I found this utility to be of unprecedented benefit and it is possibly the nearest the Dragon will ever get to Desk Top Publishing.

Simon Jones



Winners and Losers

Every month
Gordon Lee will
look at some prize programming

BY an odd coincidence, this month's postbag contained several requests for further information on determining if a given number is an exact power (square, cube, or whatever). Typical of these was the note from Neil Davidson of Akrotiri, who askes: "Could you please indicate on your answer page the tests required to do this. I tried everything I could think of as well as the method you put on the *Winners and losers* page of the September issue."

We'll get back to this problem shortly, but first a few remarks about the competition itself. A large number of competitors — in fact, well over half — did a bit of algebra on the original information given in the puzzle and reduced the problem to finding the solution to the equation:

$$z = 6x^2$$

If we had blind faith in the infallibility of the computer, **listing one** would be an example of a program designed to solve the equation, and the answer produced by it, $x=750$, $y=150$, would lead us to assume that this is the correct solution. This was the case with quite a number of readers! The fact that this method fails to provide either of the examples given in the problem itself should have set the warning bells ringing!

Several solutions were received with line 40 amended to:

```
40 IF ABS(Z-INT(Z)) <0.00001 THEN
   PRINT X,Z
```

This alternative, which was intended to take into account the mathematical inaccuracies inherent in calculating roots, did indeed produce the two lowest solutions, but then jumps to the $x=750$, $y=150$, completely missing out the required answer. The reason is not hard to discover — and reveals one of those quirks which can so often cause problems with programming generally. We have observed and accepted that the computed values of certain roots can be a very tiny bit out, and the modified line 40 shown above was designed to take account of this. But the use of this line makes one big assumption — which is not always correct! If you want to test your logical powers, try to find the flaw in line 40 before reading further.

The assumption is that the computer root is a value slightly *greater* than the correct value. What if it is that tiny bit *under*? This is the case with the correct solution (which was why that value was chosen!). In the case of the correct solution it is necessary to test the value 157464 to determine if it is a perfect cube. Now the computed value of its cube root is 53.999999955, so the 'Z-INT(Z)' part of line 40 will give a result of almost 1 — little wonder that this result failed the test! A quick experiment (**listing two**) will reveal those prefect cubes whose cube roots are underestimated in this way. In fact, on the Dragon, of the cubes of all integers from 1 to 1000, one hundred and nineteen are computed exactly, 582 are slightly too high, and the remaining 299 are too low. The solution given in the March issue shows one method of overcoming this problem.

However, the final words this month must go to those readers who used a delightfully simple routine for performing this test — a method which can be easily adapted in other forms of testing of this type. **Listing 3** shows it in action. The important part is in line 30 where a hefty 0.5 is added to the computed root *before* the integral value is taken. This ensures that any discrepancy will result in a positive difference, and line 40 then checks the cube of this value (by direct multiplication) with the number under test. My thanks to S P Grenard, Dave Lardner and R H Wilson, who used this method.

Listing one

```
10 X=1
20 Z=6*X*X
30 Y=Z^(1/3)
40 IF Y=INT(Y) THEN PRINT X,Y
50 X=X+1:GOTO 20
```

Listing two

```
10 N=1
20 C=N*N*N
30 R=INT(C^(1/3))
40 IF R<N THEN PRINT N,R
50 N=N+1:GOTO 20
```

Listing three

```
10 X=1
20 Z=6*X*X
30 Y=INT(Z^(1/3)+0.5)
40 IF Y*Y*Y=Z THEN PRINT X,Y
50 X=X+1:GOTO 20
```

Communications

Write down your problem on the coupon below (make it as brief and legible as possible) together with your name and addres and send it to Communication, 12/13 Little Newport Street, London WC2H 7PP.

Problem . . . .

. . . .

. . . .

. . . .

Name . . . .

Address . . . .

. . . .

. . . .

Adventure Contact

To help puzzled adventurers further, we are instituting an Adventure Helpline — simply fill in the coupon below, stating the name of the adventure, your problem and your name and address, and send it to Dragon User Adventure Helpine, 12/13 Little Newport Street, London WC2H 7PP. As soon as enough entries have arrived, we will start printing them in the magazine.

Don't worry — you'll still have Adventure Trial to write to as well!

Adventure . . . .

Problem . . . .

. . . .

Name . . . .

Address . . . .

. . . .

Adventure Contact

Adventure: *Tanglewood*
Problem: I have got Peabody, the gold coin, the silver coin and made some rock cakes but I don't know how to get Foghorn's specs. I know you need the fishing rod, but where is it? PS. I can help with Microdeal's *Williamsburg*.
Name: Andrew Glover
Address: 13 Gordonstoun Crescent, Orrell, Wigan, Lancs. WN5 8NZ.

The outside track

Philip Scott conjours more disc space from 'nowhere'

THE program described here was originally written to recover tracks on a DragonDOS format disc which became corrupted due to oxide coating on the pins of the WD2797 controller chip. Luckily, the directory was intact and no data was lost. The program, quite simply, uses the same technique as DSKINIT to format just one track on one side of a disc (ie 18 sectors). After using the program, I realised that it could also be used to add an extra track (or two) on a disc! (Hands up those who couldn't use an extra 4.5K or 9K).

Caution

Perhaps a cautionary word is in order here. While disc drives have a limit 'switch' to indicate track zero, which stops any further movement, the limit stop after 40 (or 80) tracks is mechanical, and does not provide any feedback signal, hence the noise when the head tries to go beyond the limit. While most drives (all I have seen) will allow the heads to move beyond specification (by one or two 'tracks' for 40 track drives and between two and four 'tracks' for 80 track drives), neither *Dragon User* nor I can offer guarantees that the drive will not suffer if you repeatedly cause the head carrier to strike the limit stop. Similarly, this area is not certified by the disc manufacturers.

Now, for those of you brave (or stupid?) enough to wish to go on, the details. **Figures 1 and 2** give familiar assembler listing and memory dump. Looking at the assembler listing, it is in three parts — a control program, a subroutine (SETDAT) to set up the 555 bytes of track data and write the track, and some fixed data and write, and some fixed data to control the formatting.

When the control program is entered, it restores the disc heads to track zero, ensuring that the heads are in a known position (and preventing the limit stop being reached accidentally). SETDAT is then called to set up the data, using the track number which was put in memory location $E9 and the 'side' information in $F3. After the data is set up, SETDAT steps the heads to the specified track position and then returns to the control module via the 'write track' function of the DOS low level disc access procedure. Finally, the control program reads each sector to check for errors before returning to Basic. If an error occurs, the program aborts at that point and returns an error code in memory location $E6.

While re-formatting a faulty track and adding an extra track are identical to the program of **figure 1**, the properties of the track are slightly different. A re-formatted track is already 'known' but may be a corrupted directory track, while a new track has to be added in to the 'known' tracks and the valid directory updated. I have therefore included two Basic programs. **Figure 3** provides the 'repair' facility, while **figure 4** adds an extra track. Both programs prompt for the information needed.

To use the facility, type in the programs of **figures 3** and **4** and save to disc. Then use

CLEAR 200,31999

to set the end of memory, load the machine code using the data in **figure 2** and save to disc with

SAVE "WRITRK",32000,32175,32000

(If you use a different name, remember to change the Basic programs accordingly).

Memory reserved

I shall explain the program of **figure 4**, which should enable the other program to be followed as well. The program starts by reserving memory from 32000 ($7d000) and loads the file "WRITRK.BIN" into that area. It then asks which drive the disc is in and SPREADs the first directory sector to extract the number of tracks and sectors per track. The IF statement in line 150 is then used to abort the run if the DOS is not DOSplus and the disc format is 80 track

Listing one

```
0000                    ********************************
0000                    * Track recovery program
0000                    ********************************
00EA     00EA                  ORG    $EA
00EA                    DSKCOM RMB    1
00EB                    DSKDRV RMB    1
00EC                    DSKTRA RMB    1
00ED                    DSKSEC RMB    1
00EE                    DSKBUF RMB    2
00F0      B7CC          COPX2U EQU    $B7CC
00F0      C004          DCMDPR EQU    $C004
7D00      7D00                 ORG    $7D00
7D00                           PUT    $2000
7D00   0F EA                   CLR    DSKCOM
7D02   AD 9F C004              JSR    [DCMDPR]   Restore to track zero
7D06   96 E9                   LDA    $E9
7D08   97 EC                   STA    DSKTRA     Set track number
7D0A   0F ED                   CLR    DSKSEC
7D0C   C6 12                   LDB    #18
7D0E   D7 E8                   STB    $E8
7D10   0D F3                   TST    $F3
7D12   27 02                   BEQ    SIDE0
7D14   D7 ED                   STB    DSKSEC     Sectors 19 to 36
7D16   0C ED            SIDE0  INC    DSKSEC
7D18   8D 16                   BSR    SETDAT     Set up data
7D1A   25 11                   BCS    ERR
7D1C   86 07                   LDA    #7
7D1E   97 EA                   STA    DSKCOM     "Verify"
7D20   AD 9F C004  LOOP        JSR    [DCMDPR]
7D24   25 07                   BCS    ERR
7D26   0C ED                   INC    DSKSEC
7D28   0A E8                   DEC    $E8
7D2A   26 F4                   BNE    LOOP
7D2C   5F                      CLRB
7D2D   D7 E6            ERR    STB    $E6
7D2F   39               LC47C  RTS
7D30                    ********************************
7D30                    *  Set up track data
7D30                    ********************************
7D30   CE 7DAF          SETDAT LDU    #BUFFER
7D33   DF EE                   STU    DSKBUF
7D35   8E 7D88                 LDX    #TRINDT
7D38   31 1F                   LEAY   -1,X       Sector sequence number
7D3A   C6 0C                   LDB    #12        12 bytes from TRINDT to buffer
7D3C   8D 34                   BSR    LC513
```

Listing two

32000	15	234	173	159	192	4	150	233	=	1160
32008	151	236	15	237	198	18	215	232	=	1302
32016	13	243	39	2	215	237	12	237	=	998
32024	141	22	37	17	134	7	151	234	=	743
32032	173	159	192	4	37	7	12	237	=	821
32040	10	232	38	244	95	215	230	57	=	1121
32048	206	125	175	223	238	142	125	136	=	1370
32056	49	31	198	12	141	52	142	125	=	750
32064	148	198	6	141	45	134	1	214	=	887
32072	236	237	193	214	243	231	192	230	=	1776
32080	164	237	193	167	192	198	18	141	=	1310
32088	25	109	162	38	225	198	3	141	=	901
32096	17	92	215	234	173	159	192	4	=	1086
32104	37	197	198	5	215	234	110	159	=	1155
32112	192	4	126	183	204	0	18	9	=	736
32120	17	8	16	7	15	6	14	5	=	88
32128	13	4	12	3	11	2	10	1	=	56
32136	53	78	78	8	0	0	3	246	=	466
32144	252	31	78	78	7	0	0	3	=	449
32152	245	254	1	247	78	20	78	78	=	1001
32160	11	0	0	3	245	251	0	229	=	739
32168	247	23	78	78	0	78	78	32	=	614

Listing three

```
100 CLEAR 200,&H7CFF : CLS : PRINT "DISK TRACK REPAIRER"
110 LOAD "WRITRK.BIN",&H7D00
120 ' *** GET REQUIRED DATA ***
130 INPUT "WHICH DRIVE";DR
140 INPUT "WHICH TRACK";TR
150 INPUT "WHICH SIDE (0/1)";S
160 ' *** SET UP DRIVE, TRACK AND SIDE VALUES ***
170 POKE &HEC,DR : POKE &HE9,TR : POKE &HF3,S
180 ' *** CALL REPAIR ROUINE ***
190 EXEC &H7D00
200 IF PEEK (&HE6) THEN PRINT "ERROR CODE";
    PEEK(&HE6);"HAS OCCURRED"
210 END
```

Table 1

Value DRAGONDOS	New / Old	SUPERDOS E6
$DD28 (+$1D28)	$51 / $4F	$DD2F (+$1D2F)
$D16E (+$116E)	$BD / $B4	$D194 (+$1194)
$D02E (+$102E)	$E8 / $A0	$D057 (+$1057)
$D02F (+$102F)	$22 / $24	$D058 (+$1058)

double sided, as both DragonDos and SuperDos contain coding which will not allow the track to be used from Basic (DOSplus will handle all formats as direct result of its more rigorous error checking).

There is also a second problem with DragonDos and SuperDos which prevents SREAD and SWRITE being used to access the extra track on a single sided 80 track disc, though files will still be placed there successfully. (There is DOS patch below to overcome both these problems without affecting the error two checking significantly).

The disc format data is then updated to add the extra track and written back to the directory. While this is cosmetic for DragonDos or SuperDos, the error checking in DOSplus will generate an error message if any attempt is made to access the extra track without this update. Next, the necessary track and side data is set up and the 'repair' program called to write the 18 sectors on the first side of the disc. If the disc is double sided, the value for the second side is set up and the routine called again.

The final action is to set up a zero length file and alter the directory entry to allocate the new track and the KILL the file to update the directory sector bit map.

By using this technique to update the bit map, the extra space is immediately included by FREE and DIR, with the exception that unpatched versions of DragonDos or SuperDos and versions of DOSplus before 3.0 will not show the extra space for 80 track double sided discs. Later versions of DOSplus and patched DragonDOS or SuperDOS will only show up to two tracks extra.

Listing four

```
100 CLEAR 500,&H7CFF : LOAD "WRITRK.BIN",&H7D00 : CLS
110 PRINT "DISK EXPANDER" : PRINT : INPUT "WHICH DRIVE";DR
120 ' *** GET TRACK AND SECTOR VALUES ***
130 SREAD DR,20,1,A$,B$ : TR = ASC(MID$(B$,125,1)) :
    SC = ASC(MID$(B$,126,1))
140 ' *** ABORT IF NOT DOSPLUS AND DISK 80 TRACK DOUBLE SIDED ***
150 IF PEEK(&HC003) <> 44 AND TR > 79 AND SC = 36 THEN CLS :
    PRINT"DISK FORMAT AND DOS NOT","SUITABLE FOR THIS ACTION" : END
160 ' *** UPDATE DISK FORMAT INFORMATION ***
170 MID$(B$,125,1) = CHR$(TR+1) : MID$(B$,127,1) = CHR$(254-TR) :
    SWRITE DR,20,1,A$,B$ : CLOSE
180 ' *** SET U SIDE 1 VALUES ***
190 POKE &HEC,DR : POKE &HF3,0 : POKE &HE9,TR
200 ' *** CALL REPAIR ROUINE ***
210 EXEC &H7D00 : IF PEEK(&HE6) THEN 420
220 ' *** SELECT SIDE 2 IF NEEDED ***
230 IF SC = 18 THEN 280
240 POKE &HF3,1
250 ' *** CALL REPAIR ROUTINE ***
260 EXEC &H7D00 : IF PEEK(&HE6) THEN 420
270 ' *** CREATE FILE ***
280 F$ = CHR$(48+DR) + ":ZZYYXXZZ.QQQ" : CREATE F$
290 ' *** FIND DIRECTORY ENTRY ***
300 FOR I = &H6BD TO &H7F0 STEP 31 : IF PEEK(I) <> 0 THEN 320
310 NEXT I : PRINT "UNEXPECTED ERROR - FILE NOT AVAILABLE" : END
320 R = PEEK(I+29) : FS = INT(R/10) : RC = 25*(R-10*FS)+1 : FS = FS+3
330 SREAD DR,20,FS,A$,B$ : TS = TR*SC
340 IF RC < 128 THEN 370
350 MID$(B$,RC-116,3) = CHR$(INT(TS/256)) + CHR$(255 AND TS) + CHR$(SC)
360 GOTO 390
370 MID$(A$,RC+12,3) = CHR$(INT(TS/256)) + CHR$(255 AND TS) + CHR$(SC)
380 ' *** WRITE DIRECTORY SECTOR ***
390 SWRITE DR,20,FS,A$,B$
400 ' *** KILL FILE AND RETURN SECTORS ***
410 KILL F$ : END
420 PRINT "ERROR CODE";PEEK(&HE6);"HAS OCCURRED" : END
```

Endstop

One final comment, both DragonDOS and SuperDOS move the head carrier towards the limit stop when attempting to recover from an error.

This should be taken into consideration if you are thinking of adding TWO tracks to the disc format. (Need it be said that DOSplus 3.0 does not suffer from this problem?)

Patch data: Any DragonDOS/SuperDOS owners with EPROM programmer capability can get access to tracks 80 and 81 by applying the changes shown in **Table 1**.

The first change allows SREAD/SWRITE to access the two extra tracks, the second includes these in the FREE space count and the last tow allow files to be added and KILLed successfully on these tracks. (Indeed, the last change overcomes the ?IV ERROR problem in DragonDOS/SuperDOS with 80 track double sided discs).

Big for its size

Ken Smith looks at the Cumana 40T disc drives in the light of experience

I have had the Cumana 40T disc drives for some time now because problems at home prevented me from reviewing it, but this is not such as bad a thing as it gives one a longer perspective on using the machinery. I have included a 'review' of the two DOS systems with the disc drives, so that prospective users will know what they are up against.

The main unit is of a solid construction having a metal case and built in power supply. It comes in a cream slightly crinkled finish and almost matches the Dragon 32. The drives are mounted side by side and marked A and B, a little confusing since the DOS refers to them as 1 and 2. There is no spring flap door (as on Dragon drives), Cumana preferring to opt for a spring loaded lever which effectively bars the disc slot when in the run position. This lever also brings the drive hub into contact with the disc, avoiding the problem of the operator trying to remove a rotating disc. It also means that you do not have to rely on a little spring to eject your floppy as you can actually grip the edge of the disc while it is still in position. Each drive has a red indicator light which glows whenever the drive is in use. To the rear is an illuminated main switch and the ribbon cable to connect to the DOS cartridge.

The size of the unit is probably its biggest fault. Having a footprint equalling that of the computer is makes it difficult to find room for other items. With this in mind and, I tried to position my monitor on top of the drive. The result was a failure to read the directory properly on drive 2. Once the monitor was removed all was well again. The extra weight must have distorted the base and interfered with the free movement of the read/write head. I now have a three tier system so that nothing stands directly on top of anything else.

In operation the system has been faultless. It is perhaps a little noisy especially when it is asked to do something it cannot do, like find track 42, but extremely reliable. As far as speed is concerned it will load a 28K file (the size of most good games) in 5-10 seconds. I know speed is always relative but anyone who got used to this system would fall asleep waiting for a Commodore disc.

To sum up, a great unit slightly spoiled by its layout. It would have been better as either two separate drives or stacked one on top of the other, preferably the first as this would give greater flexibility. The wide availability of discs and the low price make this 5.25 inch drive very attractive. As far as I know they are available, but Cumana no longer supply the DOS cartridge.

Cumana DOS

The Cumana Dos is a large, ventilated cartridge, which plugs into the side of the computer. It fits very tightly into the cartridge slot. There being no support legs to the rear of the case, I assumed that this tightness was to avoid the possibility of a program crash caused by movement at the edge connector. At the rear is the socket for the disc drive ribbon cable. There is no cartridge bus extension, so use of the DOS precludes use of any other cartridge.

The operating system is contained on a single eprom in one of two sockets. The second socket remains a mystery.

The system is supposed to be compatible with Dragon Dos however, this compatibility does not extend to machine code programs. A Dragon Dos disc will read correctly but when it comes to writing, using a Dragon Dos program, then it is something close to a machine gun simulation followed by an error message. Recognising this most companies have produced Cumana Dos versions of prgrams that use a write routine. Some features of this system, that are hard to find elsewhere, are a significant advance. The COPY command routine has been enchanced to allow you to copy to and from cassette and there is an SCOPY routine which allows you to copy a file from one disc to another using a single drive.

The system works extremely well until you need to change discs in the middle of a program. The problem here is that the contents of the directory track are stored in the buffer and it is the buffer that is accessed before reading or writing, not the directory. This may improve the access time but if you have changed discs the end result could be a total loss of all data stored on the disc. The same applies to data read from disc. Even if the data on the disc has been changed, that in the buffer remains the same until it either overflows or is cleared. As with most bugs there are ways round them if you know they are there. I just made a point not to write any more Basic programs that required changing discs.

A fifty page booklet was supplied with the system. This takes you from wiring the plug on your drive unit to programming for disc access. There is a glossary of Cumana Dos Commands and a list of error codes. The manual is fine as far as it goes which is not far enough. For instance there is no mention of how to build up a random access file system. The chapter on disc structure is all of half a page long.There is no memory map for the machine code programmer.

To summarise the system although not perfect it was reliable and the problems were at least consistant. Production ceased about a year ago.

SuperDOS

It take about five minutes to install the Superdos chip to a Dos cartridge and requires only a small Phillips head screwdriver and a pen knife or similar tool to gently prise the ROM chip from it's holder. The Superdos is now firmly placed in the newly vacated socket, and the unit reassembled. What differences you then experience will depend on your original system. Since my host cartridge was a Cumana then I will concentrate on that.

The first thing you notice is that the old Cumana title page has gone and is replaced by the standard Dragon title screen with one additional line announcing that Superdos is installed. Also gone is the SCOPY command which enabled you to copy a file from one disc to another using a single drive, as has the capability to copy a tape file to disc. Superdos keeps a backup directory on track 16 as does Dragon Dos. This results in a safer disc but a loss of 4608 bytes of file space when compared with Cumana. The system read the disc directory (not the buffer) before accessing the disc, so that a change of discs if picked up before the disc can be corrupted. However it is still advis able to close all files before changing discs. The close routine has been improved to the extent that all open files on a named drive can be closed but not individual files. The latter being a desirable feature is ruled out by the amount of ROM space needed and the need to maintain compatibiliy.

Dragon Dos compatibility is greatly improved. Flex, OS9 and BASIC42 all work with no alteration as does most of the better commercial software. However systems that try to by-pass some of the housekeeping and verification routines do run into a little trouble. The routines being different only the start addresses can be relied upon to be the same. Most such problems can be cured by a one or two byte patch. Mike Kerry (who also gave us Alldream etc.) coded this system and can usually help.

Running in Basic presents few surprises. Except for those commands that have now disappeared, the syntax is identical. One endearing feature is the way Superdos closes all open files whenever it encounters an END command. The more disciplined amongst us will not have encountered the problems caused by forgetting to close all open files before ending a program. The rest get an error report because too many files were open.

There are some idiosyncracies with the system, like the way it automaticallly closes a file if it was loaded using LOAD but does not close if you use RUN 'filename'. Also continually CREATEing files and then KILLing them seems to gradually fill up the disc (especially on drive 2). This make it necessary to copy all files onto a new disc in order to maximise storage space. This is needed infrequently, but it would have been nicer not to have it at all. The Superdos ROM can be fitted to Dragon or Cumana cartridges and comes as standard equipment in the P.N.P. Communications controller. It is a much improved version of Dragon Dos and with a Cumana system it is worthwhile both for the ease of use and compatibility.

Write: ADVENTURE

***Pete Gerrard** gets everything except Legless*

AFTER a minor interruption (or should that be interaction) last month with a look at some possible adventure ideas, back to programming this time around and that wretched word 'all'. The use of this word (as in GET ALL and DROP ALL) seems to be taken for granted in adventures these days, so, having covered DROP two months ago, for most of the rest of this article we'll be looking at GET. No other verbs will be covered, rude or otherwise, since I refuse to break new ground and be the first adventurer to have an EXAMINE ALL routine ... don't all write in at once!

The essence of the GET or TAKE command is that you are attempting to take possession of an object. In the average adventure game there will be many objects that can be carried about by the player, but equally so there will be many that cannot, for one reason or another.

Perhaps they're too heavy, or they're just intended to be part of the scenery and to help in setting the scene for the player. One wouldn't expect to be able to carry a mountain, for example. Thus our GET ALL routine needs to consider this. Do we slavishly go through every object at every location, like this:

Mountain — You can't carry the mountain.
Flashlight — Taken.
Rucksack — Taken.
Troll — You cannot possibly take the troll.

Or do we use a bit of common sense and ignore those objects that cannot be carried anyway? We opt for using common sense. Apart from the obvious fact that it makes the programming easier, and also makes it take up less memory, I'm sure it would be an irritant to the player if they had to wade through vast reams of text every time they entered a GET ALL command, just to find out what they had actually managed to pick up.

Of course, one cannot just have a GET ALL and ignore the individual command GET object, so we'll start with the latter example first and build up from there.

If you look at figure one you'll see a fairly conventional GET object routine, but in order to make sense of it (as was the case with the DROP command two months ago) you'll need to know a few things about the variables being used, and about the game itself.

The example listing is taken, albeit in slightly amended form for clarity, from the same game as the DROP routine, but just to refresh your memory if na=24 then we're talking about the 24th noun word (TENNIS) and if na=26 then we're referring to the 26th noun word (RACKET). As both of these refer to the same thing, and two words are used only as a convenience to the player, then if the word TENNIS was entered we convert it into the word RACKET, since the game itself always refers to object number 26 in preference to object number 24.

Another familiar object is object number 61, the guide dog, which allows the player to move through the cave network section of the game without the usual tedium of finding a light source. There is a flashlight in the game, but it's only there as a diversion: it doesn't work, and it never will work, but it might irritate a few players as they search endlessly for some way of repairing it!

An unfamiliar one, not given special treatment in the DROP routine is object number 12. This is a stout stick, and has two purposes in the game. One is to play pool with before the player cuts it in two with his knife, and the other is to wedge open a grate that continually falls shut if anyone attempts to go under it without first wedging it with something. Thus if the player gets the stick while it's in the location with the grate and is being used to wedge it, then the grate falls shut with a clang.

Finally, we have object number 31, who can be found in locations 12 to 17. I say who rather than which, because object number

Listing one

```
2450 REM initial get routine
2451 IF (cp>11 AND cp<18) AND na=31 AND ll>6
THEN mess=198:GOTO 5995
2452 IF na=24 THEN na=26
2454 IF ob(na)=-1 THEN mess=152:GOTO 5995
2456 REM object numbers that cannot be taken:
 trolls, that sort of thing
2458 IF ct=1 AND ob(na)=cp THEN mess=134:GOTO
 5995
2460 IF ob(na)<>cp THEN PRINT"I can't see tha
t.":GOTO 10
2462 IF na=61 AND db=1 THEN pd=0:lo=1:PRINT"T
he dog decides to follow you.":ob(61)=-1:zz=z
z+1:GOTO 10
2464 IF na=61 THEN mess=204:GOTO 5995
2466 IF na=12 AND cp=3 AND gw=1 THEN mess=60:
og=0:GOSUB 5990:gw=0:p(3,2)=0
2468 IF zz<4 THEN PRINT"Okay, "ob$(na)" taken
.":zz=zz+1:ob(na)=-1:GOTO 10
2470 mess=61:GOSUB 5990:FOR i=1 TO nn:IF ob(i
)=-1 THEN ob(i)=cp:GOTO 2474
2472 NEXT
2474 ob(na)=-1
2476 GOTO 10
```

Listing two

```
2450 IF na$="all" THEN 2480
2451 IF (cp>11 AND cp<18) AND na=31 AND ll>6
THEN mess=198:GOTO 5995
2452 IF na=24 THEN na=26
2454 IF ob(na)=-1 THEN mess=152:GOTO 5995
2456 REM object numbers that cannot be taken:
 trolls, that sort of thing
2458 IF ct=1 AND ob(na)=cp THEN mess=134:GOTO
 5995
2460 IF ob(na)<>cp THEN PRINT"I can't see tha
t.":GOTO 10
2462 IF na=61 AND db=1 THEN pd=0:lo=1:PRINT"T
he dog decides to follow you.":ob(61)=-1:zz=z
z+1:GOTO 10
2464 IF na=61 THEN mess=204:GOTO 5995
2466 IF na=12 AND cp=3 AND gw=1 THEN mess=60:
og=0:GOSUB 5990:gw=0:p(3,2)=0
2468 IF zz<4 THEN PRINT"Okay, "ob$(na)" taken
.":zz=zz+1:ob(na)=-1:GOTO 10
2470 mess=61:GOSUB 5990:FOR i=1 TO nn:IF ob%(
i)=-1 THEN ob%(i)=cp:GOTO 2474
2472 NEXT
2474 ob(na)=-1
2476 GOTO 10
2480 FOR i=1 TO nn
2482 IF ob(i)=-1 THEN 2498
2483 IF ob(i)<>cp THEN 2498
2484 REM object numbers that cannot be taken:
 trolls, that sort of thing
2485 IF i=12 AND cp=3 AND gw=1 THEN mess=60:o
g=0:GOSUB 5990:gw=0:p(3,2)=0
2486 IF i=61 AND db=1 THEN pd=0:lo=1:PRINT "T
he dog decides to follow you.":zz=zz+1:GOTO 2
496
2488 IF i=61 THEN mess=204:GOSUB 5990:GOTO 24
98
2490 IF zz<4 THEN PRINT "Okay, "ob$(i)" taken
.":zz=zz+1:ob(i)=-1:GOTO 2498
2492 mess=61:GOSUB 5990:FOR j=1 TO nn:IF ob(j
)=-1 THEN ob(j)=cp:GOTO 2496
2494 NEXT j
2496 ob(i)=-1
2498 NEXT i:GOTO 10
```

31 is a person rather than a thing. It is our old friend Legolas the elf, but as the game progresses the valiant elf pays increasingly frequent visits to locations 12 to 17. These are areas of a pub, so sadly Legolas rapidly degenerates into Legless the elf, and a someone pointed out it would be nice to have a response for someone typing in GET LEGLESS! An advertisement for my local pub seemed reasonable, so that's what message 198 in line 2451 is all about. The variable ll is used to keep track of the elf's visits to the pub, and he is switched from Legolas to Legless after he's had six drinks.

Line 2452 now becomes self-apparent, while line 2454 caters for the possibility of the player trying to get an object that he is already carrying. If the object's current value is -1 then this is indeed the case, and we use message 152 and the routine at5995 to inform him of this fact.

Line 2456 has not been included in its original version, because to do so would necessitate many paragraphs of explanation. Basically, it's a long list of IF NA=1 OR NA=2 OR etc., naming all the objects which, for one reason or another, cannot possibly be carried by the player: mountains, trolls, that sort of thing.

Line 2458 is another special one, because it handles the situation of the player trying to get something on the ground when the variable ct has been set. This indicates that the player is currently half way up a tree, and message 134 is a sarcastic one about the problems involved in reaching the object when you're twenty feet off the ground.

In line 2460 we check to see that the object is actually in the same location as the player. If it's value is not equal to cp, the current position of the player, then it isn't in the location, so we just print up a simple message to that effect and return to our control line, line 10.

In line 2462 we deal with the guide dog. The variable db is used to see if the dog has been given a bone. If he hasn't then he is unlikely to co-operate, because he's hungry, fed up, and mean. But, if he has (a loathsome pun gives you a bone of contention, which you can then give to the dog: it's a strange game!) then the pitch dark variable pd is set to zero, the lights on variable lo is set to one, meaning that we can now traverse the dark locations of the game. A simple message about the dog is then printed up, before we put it in the player's possession and increment the number of objects being carried variable, zz. Why zz? I haven't the faintest idea, just a whim.

Line 2464 then sorts out the player trying to get the dog before he's given it a bone, while line 2466 concerns itself with the stick, location 3 (where the grate is), and whether or not the grate has been wedged (gw=1 indicating that it has). All this being so then we reset the open grate variable og to zero, print a message about the gate falling shut, and cut off the route south from location 3.

Line 2468 checks to see how many objects are being carried. If the current number is less than 4 then we simply add the object to the player's crop of goodies, tell him that this is what we've done, increment the number of objects being carried variable, zz, and return to our control line 10. However, if the player's carrying more than this then as he attempts to pick up the new object he proceeds to fumble about and drop one of the other ones, which is what the routine in lines 2470 to 2474 is all about. The first object that the player is found to be carried is dropped to the floor, after message number 61 has been printed up (something about fumbling and dropping something), then in line 2474 we allow the player to get the object that he was after in the first place.

Finally, we retreat from the routine in line 2476. Simple? Mais oui!

GET ALL takes much the same sort of form, but first of all we must re-enter line 2450 as shown. This then takes us off to line 2480, and from lines 2480 to 2496 of **figure two** we're concerned with trying to GET ALL the objects in sight.

Line 2480 sets up our loop to start going through each of the objects in turn (there being nn of them), and in line 2482 we ensure that if an object is already being held then we merely continue around the loop. Similarly, in line 2483, if an object isn't in the player's current location we ignore it and carry on with the next step of the loop.

In the next few lines we deal with all those objects that the player cannot possibly get, and our special ones: the stout stick and the guide dog. Finally, in lines 2492 and 2494 we have the routine for working out which object gets dropped when the player is attempting to pick up more than he can actually carry. This could, of course, result in many messages being printed up about objects being fumbled and dropped to the ground, but if it's good enough for Infocom then it's good enough for me!

And that is one, relatively straightforward, way of performing a GET ALL routine. You may care to amend it to include something along the lines of 'There isn't anything here to get', just in case a player might hopefully try and get everything when there isn't anything there.

That's it for this month. Bye for now.

Pete GERRARD'S ADVENTURE TRAIL

Tolkien Tolkien Tolkien. There, is everybody happy now, particularly editors who go around inserting strange comments about Scotsmen in make up and mini skirts into the hallowed paragraphs of this beloved adventure trail? It's hard to write seriously when you are constantly aware of two beady editorial eyes beaming their attention down on your every utterance, and one can only try one's best ... (*We bin givin' him a Tolkien to ...*)

I received a strange epistle the other day, from none other than the legendary wizard Strombrigner the Grey: regular readers will know of his parents' dyslexia, resulting in his rather unusual name. Together with his accomplice, Dimli Gloing, dwarf of astonishing thirst, they have united their unique powers on many an occasion.

Master Peter (began the epistle), you may be interested in one of our exploits for your annals. Whether others will find it of interest it is hard to say, but the story is an unusual one, and I shall leave it to your discretion as always. I know that I can safely trust your judgement in these matters.

Well, I read on, and what an astonishing little escapade they had been through! I have no hesitation in passing it on to you, it may help one or two souls struggling bravely in an adventure beyond their comprehension.

The epistle continued:

We were in Cricklewood, young Dimli and I, investigating an incident which need not concern us here. Unfortunately the matter was taking longer than anticipated, we had little of note in the way of funds, and we thought it best to go to the local DHSS and sign on for some emergency money. A most depressing place, one I do not wish to visit again in a hurry. Having signed on we

did at least have a reasonable amount of wedge at our disposal (it is encouraging to see how well these folk look after wizards and dwarves: precious few vacancies, let me tell you), and together we wandered off to the east and visited a most charming garden centre.

Bush

There we saw some particularly impressive looking shrubs, and in a moment of impetuous madness I was persuaded to buy some. Why, I do not know, since my talents do not lie in the direction of gardens and their contents, but there you have it.

As the afternoon wore long we spent a pleasant hour or so in a nearby sauna, where Dimli had an unfortunate accident with an iron bar. He protested to me afterwards that it merely came away in his hand, but I do not know. Still, one does not question a dwarf under such circumstances, and after leaving the sauna (and, I might add, if I were not 832 years old, some of the inhabitants of that sauna might well ... but I digress) we hopped on a bus and alighted by the side of a cheese shop.

As you know, Master Peter, I have often been one for the more illustrious cheeses of this green and pleasant land, although it was something of a surprise to find such a place as this thriving in North London. Alas, Dimli had something that can best be described as a disagreement, resulting in a fearsome blow to the unlucky Greek. On the plus side, we did gain a map for our troubles. I was beginning to think that we were getting into this in a pretty deep sort of way, and immediately ordered the reckless dwarf to hop onto a nearby omnibus.

We disembarked by the side of a chemists. Dimli saw and instantly fell in love with a truss, which he procured. Having no need for such garments, and ever one to think of the future, I acquired some travel sickness pills. As you know, my liking for travel of the train variety is not great, and one cannot help but think that mankind would be better served by the more graceful passage of the stage coach. Heaven knows, highwaymen abound on both methods of transport.

We travelled to Euston station. What a concourse! Enormous, and fearfully hot. The air conditioning is appalling, and it was with some relief that we managed to find the relative comfort of a ticket office. There, economy being the thing, we purchased a Rail Rover ticket, and at once set about looking for a suitable destination. Inverness, the magic of the highlands, the sound of the pipes, the Flying Scotsman, we boarded our train from platform six and spent the next few hours on a comfortable passage north. I was somewhat alarmed by young Dimli's insistence on visiting the buffet car at the slightest opportunity, but he has a liking for cans of McEwans, and never seems any the worse for it. I consoled myself with the occasional medicinal scotch.

On reaching Inverness, heart of the highlands and a commendably clean and tidy city, we headed south, some unknown instinct pulling us in that direction. Alas for instinct, we were met by some extremely Nasty Knights, but fortunately for us they were soon placated by a gift of the very shrubs that we had purchased earlier. A close bush with danger, eh? I shrubbed it off.

We continued moving south until we reached a clearing. A rock, clearly guarding something of interest, proved too much for my ageing back, but Dimli had it cleared in a second. There we found a key, a small, curious sort of key, which I pocketed for later use. From our extensive surveillance it was obvious that Inverness currently held nothing of further interest to us, and so it was back to the station and a hasty return to Euston. Or, at least, as hasty as the good people of the trains would allow.



Brain

Once back in London my reasoning brain told me that the key which we had found was of a variety often to be seen in Lost Property Offices, and so it was to there that we journeyed. Feeling, I might say, vaguely guilty about the whole affair. Our original enquiry, something about which I shall narrate in a further letter, was being sadly delayed, but the affair of the squashed penguin will have to wait for another time. We were onto other things, and from a steel locker in the lost property office we managed to obtain a library ticket.

It was an easy matter of deduction, even Dimli saw it, and we proceeded at once to the Walpole Memorial Library. We could have obtained many books, it is true, since surveillance seemed sadly lacking, but with that determined manner of his Dimli chose a very big book. Having obtained that we returned to Euston and, at the merest whim, set off for Alice Springs. I would have thought it a long journey by train, but I was incorrect, and it seemed that only seconds had passed before we arrived at the place. As inhospitable as I had imagined, and with nothing else to do I thought we might as well head straight back. Dimli insisted on visiting the toilet and, knowing how important it is to do these things when you do not know when the next opportunity will arise, I followed.

B . . . convenience

There was something rather odd about that gents' toilet, and I centred my gaze on the cistern. I know of old when something has been tampered with, and looking at that rusty old cistern I discovered something that might well astonish you as much as it did myself: a Holy Hand Grenade. I finished my ablutions and took the grenade with me before we once more boarded the train and concluded our peculiar journey by travelling back to Inverness.

We went to the already familiar clearing and then travelled to the east where, via a tavern (oh, how much effort went into persuading the blessed dwarf not to stay there!) and a castle we obtained some woollen garments and a spade. The former of immense use in keeping out the cold, the latter, well, we would no doubt think of something ere long.

We retreated back via the tavern (and another struggle with Dimli's thirst) to the clearing, whereupon we switched directions to confuse any who might be attempting to follow, and headed off south until we reached the base of a cliff. There, a brainwave of which you would be proud, Master Peter. We tied the garments together and set off to climb the cliff. It was not easy, my aged bones ached with silent protest, but we managed it. We noticed a cave, and we further noticed a rabbit guarding it! A rabbit, mark you. I was preparing to begin negotiations when I saw that Dimli's impatience had reached its limit. With a mighty heave the grenade found itself too close to the rabbit for the rabbit's liking, and with my eyes shielded we entered the cave.

There we found a use for the spade at last, and Dimli began to dig like an orc. It was but a matter of seconds before we had uncovered that which I knew we had been seeking all along. It was the Holy Grail.

Buffet

We returned to Inverness, and from there set off on our last train journey south back to Euston. It was just as well, I was beginning to weary of trains and their constant rocketing, buffeting motion. We were in need of rest, and from Euston we went to a padded cell, laid the Holy Grail down, and slept the sleep of the just. We knew it was a mission well completed, and if you were to publish this tale Master Peter may I respectfully suggest that you name it *The Cricklewood Incident*.

Yours as ever

Strombrigner the Grey.

Well (this is me, now), what a story, eh? I am fortunate in having such luminaries as Professor Deadrock and Strombrigner amongst my correspondents. Until the next time, adventurers everywhere! Bye for now.

The Gordon Ratio

Gordon Lee finds the Golden Ratio, but isn't satisfied with that . . .

AS programs listings go, it would be difficult to find anything shorter than this:

```
10 X=1
20 X=(X+2/X)2:PRINT X:GOTO 20
```

Yet, when run it produces a surprising result by rapidly computing the square root of 2, producing nine-digit accuracy in only four cycles of the program. The value given for X in line 10 can be any value chosen at random, it doesn't matter. In a fairly short time the result will be the same: 1.41421356. In fact, the same formula can be modified to find the square root of any value that you wish (within the mathematical capabilities of the computer). Simply replace the 2 that occurs within the brackets with the value whose square root you wish to find.

This method of using the result of a calculation to initialise a repeat of the same calculation is called 'recusion'. In a sense, the computer 'learns' from past experience, and uses this to re-calculate to a greater degree of accuracy. Another example would be to find a value that becomes its own reciprocal when 1 is subtracted. In other words a value, X, which satisfies the equation

$1/x = x-1$

By adding unity to both sides of this equation we get: $x = 1+(1/x)$, an expression we can use in the above program. Simply substitute this for the one in the listing at line 20 and re-run the program. This time, the calculation takes a little longer to 'settle out' at a constant value, but very quickly gives the result 1.61803399. This, you may recognise as the Golden Ratio, a value known to the ancient Greeks, as having certain mathematical and aesthetic properties. For example, a rectangle with sides in the ration of 1:1.61803399 has a remarkably well-balanced proportion, a fact which accounts for it being used so often in art and architecture. If a rectangular piece of paper in these proportions has a square cut from one end, the remaining peice will also be in the same proportions, and so the cutting excercise can be repeated *ad infinitum*.

This same value is also found if a division is performed with consecutive values of the Fibonacci series. This series of numbers, named after the 13th centurey mathematician, is that formed by starting with 1 and 1, and then finding each successive number by summing the preceding two. So the series will run 1,1,2,3,5,8,13,21,34...and so on. As the series progresses, any value divided by the one before it in the series will produce the golden number — the further along the series the numbers are, the more accurate will be the result of the division. A short program can easily be written to test out this calculation. The full value of the golden ratio extends, as an irrational decimal, to infinity, so only the first few digits can be checked with absolute accuracy.

Using a series, such as the Fibonacci numbers, to generate an irrational mathematical constant is not restricted to just the golden ratio. Other constants can also be produced as a direct result of a logical series. That most enigmatic value, pi, the ratio of the circumference of a circle to its diameter, can be computer from a series in a number of ways. For example, pi is equal to the continuing series:

$4(1/1-1/3+1/5-1/7+1/9-1/11+ ...)$

A simple program will demonstrate this, although this formula converges only very slowly and so is not really of any great practical use. The one above was discovered by Gottfried Wilhelm, Baron von Leibniz, during the 17th century, and it was by the use of ther, more rapidly converging series, that more accurate approximations to pi were calculated. In 1699, Abraham Sharp calculated pi to 71 decimal places. Gauss and Dase took the limit to 200 places in 1824,

Prize

THOSE who can climb to the end of Gordon Lee's series and come back again with 125 decimal places will be in line for another climb, this time up Dragon Classic **Bean Stalker**, which is being contributed to *Dragon User* this month by new software house from South Wales, **Orange Software**. So peel away there ...

Rules

When you have descended into the depths of mathematical hell climbing towards an answer, conceal it from human sight in an envelope marked MAY COMPETITION (not *MIGHT COMPETITION* or even *MAYBE COMPETITION*) with your hard-won printout and any comments or last words you wish to include, and post it to us. Tell us who you are, and where you live. Sit and wait. Hope.

But first abandon hope! You thought we'd forgotten the tiebreaker, didn't you? No, never, never ... what can we do to you this month? Be fair: no poetry, no carols, no limericks. Just let your imagination wander: if you had to climb up the beanstalk and fight the giant, armed with nothing but a monster roll of Andrex, what would make it all worth while on your return?

Unprintable replies will go to the bottom of the pile; proposals of marriage to either Gordon Lee or the Editor will be considered on the strength of purely practical considerations.

February winners

THE solutions we had varied wildly in speed and method, but some of the tiebreakers were so good we have a mind to send them to British Rail.

Anyway, getting back to the real business, the fastest logged time was from Pete Faraday of Warrington at just over 4.3 seconds. Not far away were S A Siddiqui of Acton, R H Wilson of Basingstoke (the nice gent who thinks it only takes us a month to process all your comp entries. No wonder he thinks there's a practical solution to British Rail!), D J Gray of Warrington, T. Fawcett of Hendon and T. Wilson of Northampton all followed with scores in the 8 to 10 seconds area; J. Smith of Twyford, Graham Barber of Sutton Coldfield, Peter Barker of Walsall, F J Taylor of Middlesborough, C. Hitchinson of Middlesborough, E F Stanley of Chatham, Fred Willers of Yarnfield and P. Morgan of Bristol had entries in the 10 to 30 second range, and R. Raine of Sapcote, Michael White of Salford, Steve Turner of Eccleshall, Martin Reid of Merseyside (couldn't read the district, Martin) and H. Smith of Plymouth beat Gordon Lee's own solution to the crunch by a greater or lesser margin.

Denis O'Mulloy, usually one of the stalwarts, proved that the gods take their vengeance on those who ignore the message of the tiebreakers, by creeping in at a mere 260 seconds. Still, at least he entered. Fie on those of you who didn't.

Some of the many sound solutions for getting the trains to run on time . . . move the stations closer together . . . abolish timetables . . . abolish the passengers, because they complain . . . change the timetables to match the times of arrival . . . only run trains downhill . . . make all trains non-stopping . . . run the first train of the day early so that all the late ones look early as well . . . free bar for everybody picked up more then 10 minutes late (compulsory) . . . electrifying the drivers instead of the trains . . . but the best "cause the editor to make a noise like a hysterical horse at feeding time" contribution was an 'overheard' from C. Hitchinson — "If they'd spend less time chasing shrimps for the First Class, maybe we could get started." The *Chuckie Egg* should be with you by now, C.

Solution

See opposite.

and Shanks worked to 707 decimal places in 1873. The fact that he made an error and all digits after the 528th were incorrect does not really detract from this remarkable effort.

Other series which you might like to test by means of short programs are:

1/1=1/2+1/3-1/4+1/5-1/6+ ...

which approaches the natural logarithm of 2 as its limit, and

1+1/1!+1/2!+1/3!+1/4!+1/5!+ ...

a series which gives the value *e*, the base of the natural logarithms. (The ! after the denominators in this series indicate the factorial of that number. So 5! means factorial 5, which is 5x4x3x2x1). The value of *e* can also be given by the expression:

$$(1+(1/n))^n$$

when *n* is any positive interger — the larger its value, the more accurate the result. Suprisingly, this equation is encountered in formulae for computing compound interest — an illustration of the remarkable frequency with which widely differing mathematical concepts are interrelated.

This month we have wandered around a number of examples in which the computer can be used for repeated calculations, either in recursive techniques, or as a means of calculating constants by means of various series. We began by considering a simple method of finding the square root of 2. This will produce a result to a few decimal places. For the competition this month the task is to devise a program to compute this value to an accuracy of 125 decimal places. If you have printout facilities, then send the complete computation. If not, just let us know the final five digits in this value — that is, decimal places 121 to 125 inclusive.

The Answer

This is Gordon Lee's own solution to the February competition see page 22 for results

ANSWER: The problem was to devise a 'speeded-up' routine for the multiplication of two fifty-digit numbers. If these numbers were inserted in the listing 2 as given a running time of over four and a half minutes was required to complete the computation.

A marginal improvement in this running time could be obtained by removing unnecessary spaces and REM statements, and the condensing of the program into multi-statement lines, but to show any significant increase in speed a more radical approach would be required. The main factor which limits the speed of the program given is that every digit in the multiplication has to be multiplied by every digit.

In the listing given here certain modifications have been made:

(1) The value in A$ is initially multiplied by 1 to 9 and the resulting strings stored in the array (M$)

(2) All calculations are performed on blocks of eight digits rather than on single digits as before.

Listing A

```
>
  10 TIMER=0:CLEAR 1000
  20 A$="73795304344252726633390147358974818114755211845351"
  30 B$="30658548775849541883672359302221775860324499365539"
  40 DIM M$(9):REM Set Arrays, Variables etc.
  50 S$=A$:GOSUB 290:A$=S$
  60 M$(1)=A$:FOR B=2 TO 9:Z$="":C=0:REM Compute sub-products
  70 L=LEN(A$):FOR F=L TO 1 STEP-8
  80 V=VAL(MID$(A$,F-7,8))*B+C:C=0
  90 V$=STR$(V):V$=MID$(V$,2):IF LEN(V$)>8 THEN C=VAL(LEFT$(V$,LEN(V$)-8)):V$=R
IGHT$(V$,8)
 100 IF LEN(V$)<8 THEN V$="0"+V$:GOTO 100
 110 Z$=V$+Z$
 120 NEXT
 130 S$=Z$:GOSUB 290:M$(B)=S$:NEXT
 140 L=LEN(A$+B$):Z$=STRING$(L,"0"):C=0:REM Compute sum of sub-products
 150 FOR F=LEN(B$)TO 1 STEP-1:D=LEN(B$)-F
 160 B=VAL(MID$(B$,F,1)):IF B=0 THEN 240
 170 Y$=M$(B):LY=LEN(Y$):PZ=L-7-D
 180 FOR G=LY-7 TO 1 STEP-8
 190 V=VAL(MID$(Y$,G,8))+VAL(MID$(Z$,PZ,8))+C:C=0
 200 V$=STR$(V):V$=MID$(V$,2):IF LEN(V$)>8 THEN C=VAL(LEFT$(V$,LEN(V$)-8)):V$=R
IGHT$(V$,8)
 210 IF LEN(V$)<8 THEN V$="0"+V$:GOTO 210
 220 Z$=LEFT$(Z$,PZ-1)+V$+MID$(Z$,PZ+8):PZ=PZ-8
 230 NEXT
 240 NEXT
 250 IF LEFT$(Z$,1)="0"THEN Z$=MID$(Z$,2):GOTO250
 260 PRINT Z$
 270 PRINT"Time taken:";TIMER/50;" seconds"
 280 END
 290 S=LEN(S$):N=S/8-INT(S/8):IF N<>0 THEN S$=STRING$(8-N*8,"0")+S$
 300 RETURN
```

To consider the first of these: in the course of any long multiplication much ground is repeated on each line of the operation. For instance, in B$ there are eight 5s, so in normal long multiplication it would be necessary to multiply the digits in A$ by 5 a total of eight times. The listing given here performs this task once, the resulting product being stored in the array at MS(5). The same procedure is carried out for each of the other products of 1 to 9. Once this is done, these sub-products can be accessed directly and are simply 'added' into the final product in string Z$.

The second 'improvement' is to carry out all calculations on groups of eight digits rather than just single digits. As the computer is able handle numbers of this magnitude with absolute accuracy (at least as far as the basic four mathematical functions is concerned), dealing with sets of digits in this way will speed up the operation. To prevent problems with taking groups of digits, all string variables which are being used to hold the calculation are initially increased in length to a multiple of eight characters by placing additional zeros at the left hand end. This is done in the subroutine at line 290. Any of these extra zeros still at the beginning of Z$ are finally removed before the result is printed out (lines 250 and 260).

As in the program listing on the February competition page, a series of numeric variables are used to denote the magnitude of the sequence of digits being operated upon, and hence the position within Z$ that the resulting answer must be inserted.

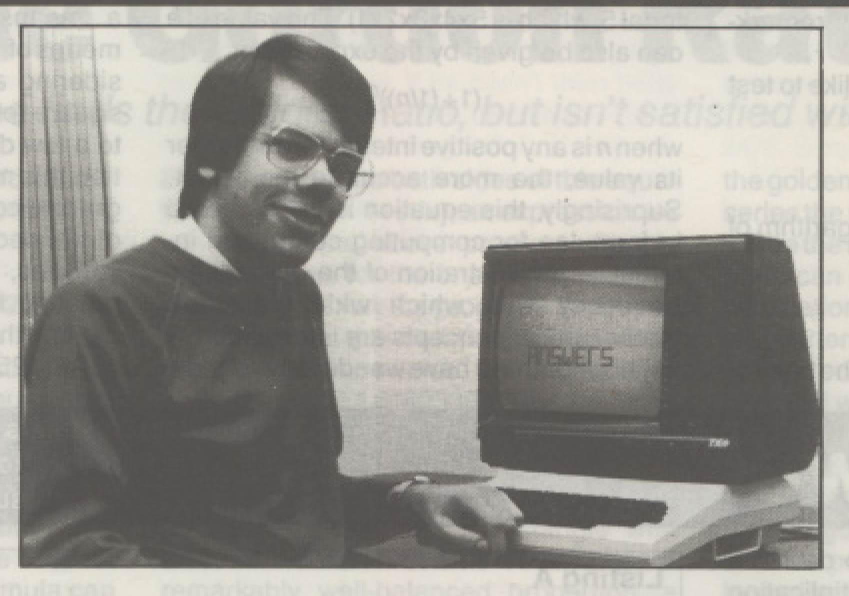
The original program running time was of 278 seconds. The listing given here computes the product in 75 seconds.

Classified

The coupon is on page 15

Dragon Answers

If you've got a technical question write to Brian Cadge. Please do not send a SAE as Brian cannot guarantee to answer individual inquiries.



Arrays to declare

I AM relatively new to programming in Dragon Basic. I am trying to declare some large arrays for data storage, but keep running out of memory. I have tried typing PCLEAR 1 which solved the problem for a while, but I would like to be able to access the memory used by graphics page one — however, typing PCLEAR 0 gives an error — can you explain?

Dougie McCourt

DRAGON Basic will not allow you to release the first graphics page (with a PCLEAR 0). However, you can do this BEFORE loading a program by typing the following:

POKE 25,6:POKE 1536,0:NEW

or with DragonDOS, change this to:

POKE 25,12:POKE 3072,0:NEW

This will give you just over 30K of free memory to use from Basic.

Serial Dragons

CAN you tell me how Dragon Data issued their serial numbers and what ranges were used for each issue of the main processor boards of the Dragon 32 and 64?

UNFORTUNATELY, I cannot give you a range of serial numbers for each board issue. Dragon Data did not issue such a list, but I do know that the Dragon 32 serial numbers were not issued sequentially, therefore it is unlikely that such a list exists!

A merge emerges

I HAVE a number of common subroutines which I wish to store on one tape, with the programs on another. However, when one file is read in it erases the routine already in memory.

How can I load the subroutines one by one into my overall program?

D W Evans
53 Wood Green
Witney
Oxon OX8 6DB

WHAT you need is a 'merge' command (which Dragon Basic does not have!). This is one of those questions which comes up fairly regularly — here's how to merge to programs together...

1) Load the first program from tape.
2) Type: ? PEEK(25),PEEK(26)
(Call these values A and B)
3) Type: ? PEEK(27)*256+ PEEK(")-2
(Call this value C)
4) POKE 25,INT(C/256): POKE 26,c-PEEK(25)*256
Where C is the value noted in step 3.
5) Load in the second program
6) Type: POKE 25, A:POKE 26,B:EXEC 33773
Where A and B are the values noted in step 2
7) Save the program with CSAVE "NAME",A

Of course, the two programs must not have line numbers that overlap! The machine code routine called in step 6 simply carries out the normal initialisation after a Basic program is loaded.

Comms board coming

I regularly get letters from readers who are interested in data communications using a modem (bulletin boards etc.). So for those of you who are interested, here is an extract from a letter I received from Jim Fuller of 42 Kitchener Road, Amesbury, Wilts SP4 7AD:

"I have designed, built, tested, and am now making available to others, an RS232 port that is both software and hardware compatible with the serial port on the Dragon 64. The main advantage of this port over others is that it can be used by comms software written for the 64 (assuming the software does not need the extra RAM or 64 specific ROM routines).

"The RS232 upgrade is contained on a small printed circuit board that fits nearly inside the Dragon's case, under the keyboard. The 7-pin DIN socket is located on the left of the machine as on the 64, and is pin for pin compatible. The expansion port is still free for use by (for example) a DOS cartridge.

"The upgrade is available for users at £30. For a further £8.50 an installation service is available from Chris Foster (Tel: 0722 790530)."

"Sorry it's such a paltry offering" says Brian, "but I answered all the letters I had." He is threatening to take another holiday if someone doesn't think of some more questions. He isn't taking a holiday on what we pay him, that's for sure ...

Roll out the Show

***Graham Smith** returns from Cardiff Airport with a few pennies in his pocket.*

THE latest Dragon show was held on Saturday 26th February and arranged by John Penn Discount Software at Rhoose Airport, just outside Cardiff.

Apart from ourselves, Orange Software, the show was supported by Computape, Prestons, Harry Whitehouse and Dragonfire Services. There were deomnstrations from the 68 Micro Group, an amateur radio group and a chap who I believe was David Makin of *Music Maker*, although I didn't have time to enquire.

This was the show we had chosen to launch Orange Software on an unsuspecting public. I put my specially purchased orange shirt.

To our left, Rick Applegate and Ted Bacciarelli from the 68 Micro Group managed to convince me to renew my membership in their group. Further along that side of the room, Andrew Hill, assisted by Tudor Davies, was selling software under the Dragonfire Services banner, most of it packaged in green!

The radio amateurs decided to move to the other end of the airport terminal to avoid interference problems. Just by the door, Harry and Verony Massey of Computape were selling Quickbeam stock — now that Dave Hitchman has called it a day, Computape have picked up his stock alongside Microdeal's.

On the other side of the room, we had Bob Preston selling the R & A J Preston range of software and some others. Sandwiched between him and Harry Whitehouse was the music man himself, Dave Makin, demonstrating his tunes. Harry Whitehouse was selling his usual range of Dragon peripherals and his famous power supply.

Finally, tucked up in the back of the room, were the organisers, Helen Penn of John Penn Discount Software.

There you have it. I thought I had better give you some idea what it was like, as not many of you turned up to see it. I think attendance was about 120, not the busiest show I have been to! However, on the bright side, we had people at our stand all day, playing the demonstration games, and quite a few of them actually bought something too. We left the show in profit, if not rich and yes, we are going to Ossett. See you there.